(NASA-CR-120560) A MAN IN THE LOOP TRAJECTORY OPTIMIZATION PROGRAM (MILTOP) Final Report (Interactive Computer Systems, Inc., Athens, Ga.) 89 p HC CSCL 22A G3/13 — N75-13898 — Unclas 06490

A MAN IN THE LOOP TRAJECTORY OPTIMIZATION PROGRAM (MILTOP).

by Juris REINFELDS

prepared by: INTERACTIVE COMPUTER SYSTEMS Inc.
150 Pinecrest Court
ATHENS, Georgia, 30601

prepared for: NASA
Dr R.N. Seitz
A&TS-COMP-CM
MARSHALL SPACE FLIGHT CENTER, Alabama 35812

submitted as: Final Report, contract NAS8-29029

Juris Reinfelds
JURIS REINFELDS

ABSTRACT

An interactive trajectory optimization program is developed for use in initial fixing of launch configurations. The program is called MILTOP for Man-In-the-Loop-Trajectory Optimization-Program.

The program is designed to facilitate quick look studies using man-machine decision combinations to reduce the time required to solve a given problem.

MILTOP integrates the equations of motion of a point-mass in 3-Dimensions with drag as the only aerodynamic force present.

Any point in time at which an integration step terminates, may be used as a decision-break-point, with complete user control over all variables and routines at this point.

Five automatic phases are provided for different modes of control: vertical rise, pitch-over, gravity turn, chi-freeze and control turn. Stage parameters are initialized from a separate routine so the user may fly as many stages as his problem demands.

The MILTOP system runs both interactively on storage scope consoles or in batch mode with numerical output on the line printer.

CONTENTS

1.0 INTRODUCTION

The Man-In-the-Loop-Trajectory-Optimization-Program (MILTOP) was designed to maximize interactive response flexibility and user transparency.

MILTOP achieves completely unrestricted response flexibility where any point in time may be selected as a decision-stop-point and any or all variables or programs may be examined or altered by the user at any decision stop point and then the integration of the trajectory may be resumed.

User transparency is achieved by choosing mnemonic names as close as possible to the mathematical names of the quantities concerned and by modular independence of the routines where each effect or force originates in a single place in the system so that only one routine (or at most two) have to be altered to change any one effect. For example any kind of drag-force may be introduced by a suitable alteration of the routine NEWDRAG. At any decision point any amount of mass may be dropped by

MLBS = MLBS − mass dropped

or any number of engines shut off by

NENG = NENG − number of engines shut off
or = number of engines remaining

MILTOP is written in the SIGMA [3] language because Sigma provides the most versatile general purpose graphics interactively on a storage tube CRT console.

Since Sigma is only available on CDC 6000 series computers at present, MILTOP was designed for easy translation to AMTRAN or FORTRAN. Special Sigma feature use was kept to a minimum and a method on how to achieve an almost interactive effect in a batch-mode FORTRAN environment (using a translation of MILTOP) is described in Section 5.1

Section 2 defines the mathematical problem and summarizes the coordinate systems and equations used.

Section 3 describes the design of MILTOP and each routine in detail. Each use of a special Sigma feature is explained in Fortran terms assuming that the user is familiar with FORTRAN

Section 4 describes how to use MILTOP and what options the user has at each point. Section 5 discusses how to adapt MILTOP to other similar problems or translate it into other languages.

Section 6 gives a sequence of worked examples to briefly show how a user might use the system

Finally Appendix A lists all names used by MILTOP in alphabetical order and Appendix B reviews very briefly those sigma features which are used in MILTOP and which may not be familiar to a user with Fortran knowledge only.

It may be argued that programming for user transparency and modularity generates more code than ~~when~~ programming for utmost efficiency of execution. This is not true for two reasons

(i) the argument is often used to defend a poorly thought-out solution whose code is unnecessarily complicated so that complications introduced in the name of efficiency actually introduce inefficiencies

(ii) if a user needs 3 runs at 5 seconds per run because he was confused by obscure code and got his input conditions wrong twice, then he has used more computing time than another user who makes just 1 run at 10 seconds per run

It is therefore a belief of the author that transparent code allows the user to see at a glance if he is solving the correct problem or not. This increases user confidence and allows him to concentrate on ~~his~~ his problem instead of spending a lot of energy struggling with the system

so that in the long run he will arrive at a solution faster or find a better solution, or both! 1.3

2.0 THE PROBLEM

The problem divides naturally into two distinct parts: the mathematical problem and the interactive system design problem. The mathematical problem is described in Section 2:1 and the interactive program design in section 2.2.

2.1 The Mathematical Problem

The mathematical problem is well known and has been solved in various ways many times before [1], [2] for example. This section summarizes the mathematical problem briefly to facilitate the explanation of the interactive computer program later.

2.1.1 Coordinate Systems

There are three basic coordinate systems of importance: the inertial geocentric system, the plumbline system and the auxiliary coordinate system with its y-axis along the centerline of the rocket.

The basic reference coordinate system is the inertial geocentric cartesian coordinate system $\hat{X}\hat{Y}\hat{Z}$ with the $\hat{Y}$ axis pointing north and the $\hat{X}$ and $\hat{Z}$ axis in the equatorial plane so that the $\hat{Z}$-axis is in the meridian plane of the launch site at launch time $t=0$.

The most interesting property of this coordinate system is that the earth rotates about the $\hat{Y}$ axis with angular velocity OMEGA = Ω_e

Next we have the (also geocentric) plumbline system obtained from the $\hat{X}\hat{Y}\hat{Z}$ system by a rotation of axis through the complement of the launch-site latitude and azimuth. The plumbline system is denoted by lower case letters $\hat{x}\hat{y}\hat{z}$ and obtained from $\hat{X}\hat{Y}\hat{Z}$ first by rotating ~~clockwise~~ counterclockwise about $\hat{x}$ through θ_1 and then clockwise about $\hat{y}$ through $A_Z - 90°$. A_Z is the launch azimuth (AZIM) and $\theta_1 = \pi/2 - \theta_0$ where θ_0 is the geodetic latitude of the launch site (LAT). Both AZIM and LAT are input constants set by the initialization routine INI

The coordinate systems are connected by the transformation matrix A but in our calculation we only need a_{12}, a_{22}, a_{32} which are stored as A12, A22, A32

$$\begin{bmatrix}\hat{x}\\ \hat{y}\\ \hat{z}\end{bmatrix} = A\begin{bmatrix}\hat{X}\\ \hat{Y}\\ \hat{Z}\end{bmatrix} \quad \text{where} \quad A = \begin{pmatrix} a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{pmatrix} = \begin{pmatrix} \sin A_Z & \cos A_Z \sin\theta_1 & -\cos A_Z \cos\theta_1\\ 0 & \cos\theta_1 & \sin\theta_1\\ \cos A_Z & -\sin A_Z \sin\theta_1 & \sin A_Z \cos\theta_1\end{pmatrix}$$

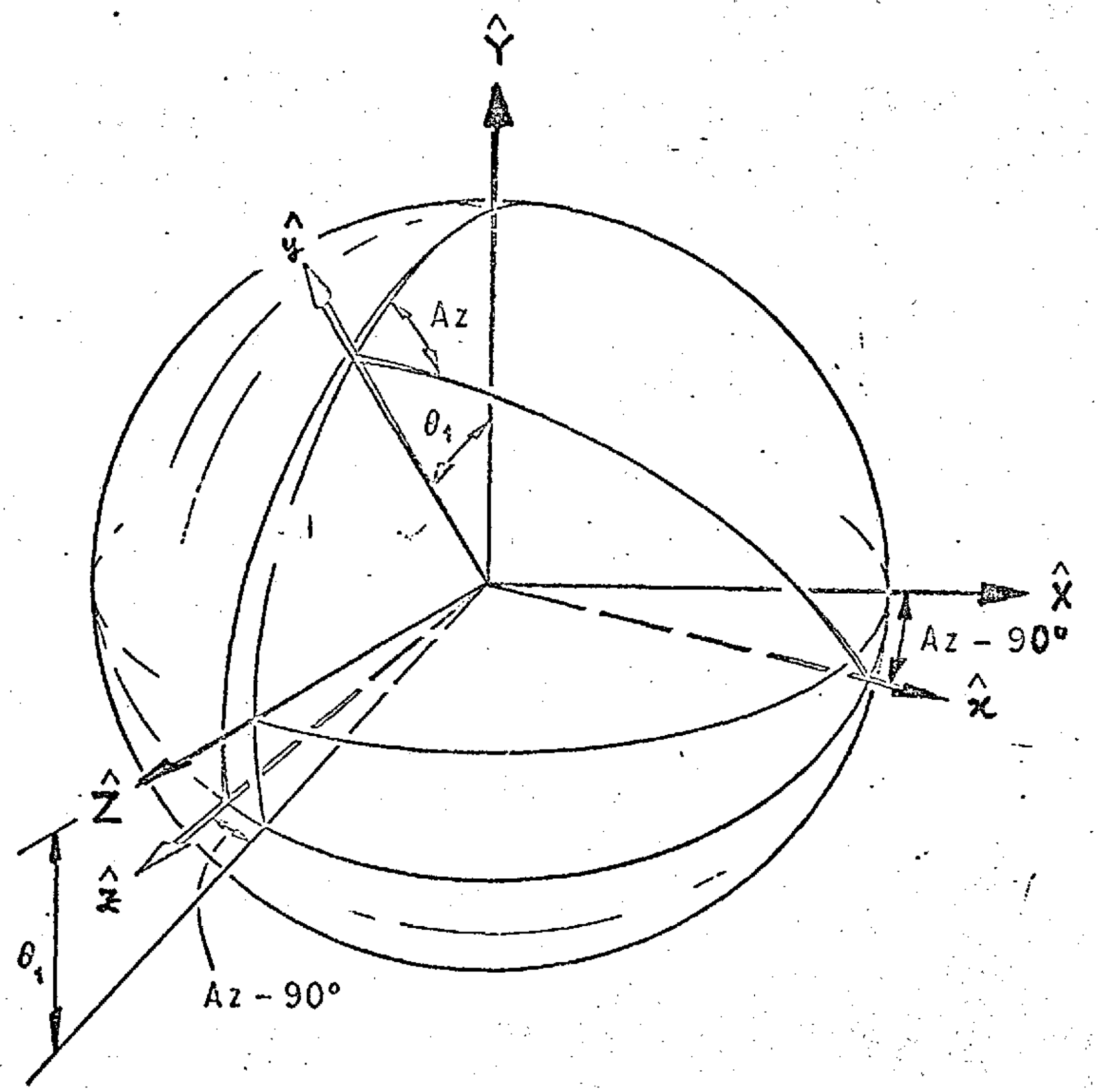


FIG. 1. PLUMBLINE COORDINATE SYSTEM $\hat{x}\ \hat{y}\ \hat{z}$

Figure 1. shows the plumbline system with reference to the inertial system. The most important property of the plumbline system is that the y-axis lies along the local vertical at launch time $t=0$ and hence, the equations of motion are most easily expressed in this system of coordinates.

The auxiliary coordinate system with reference to the plumbline system is shown in Figure 2. Its most important property is that the centerline $\hat{c}$ of the rocket lies along the "y-axis" of this system and this system is obtained from the plumbline system by a rotation through the control variable attitude angles χ_p and χ_y for χ - pitch and χ - yaw.

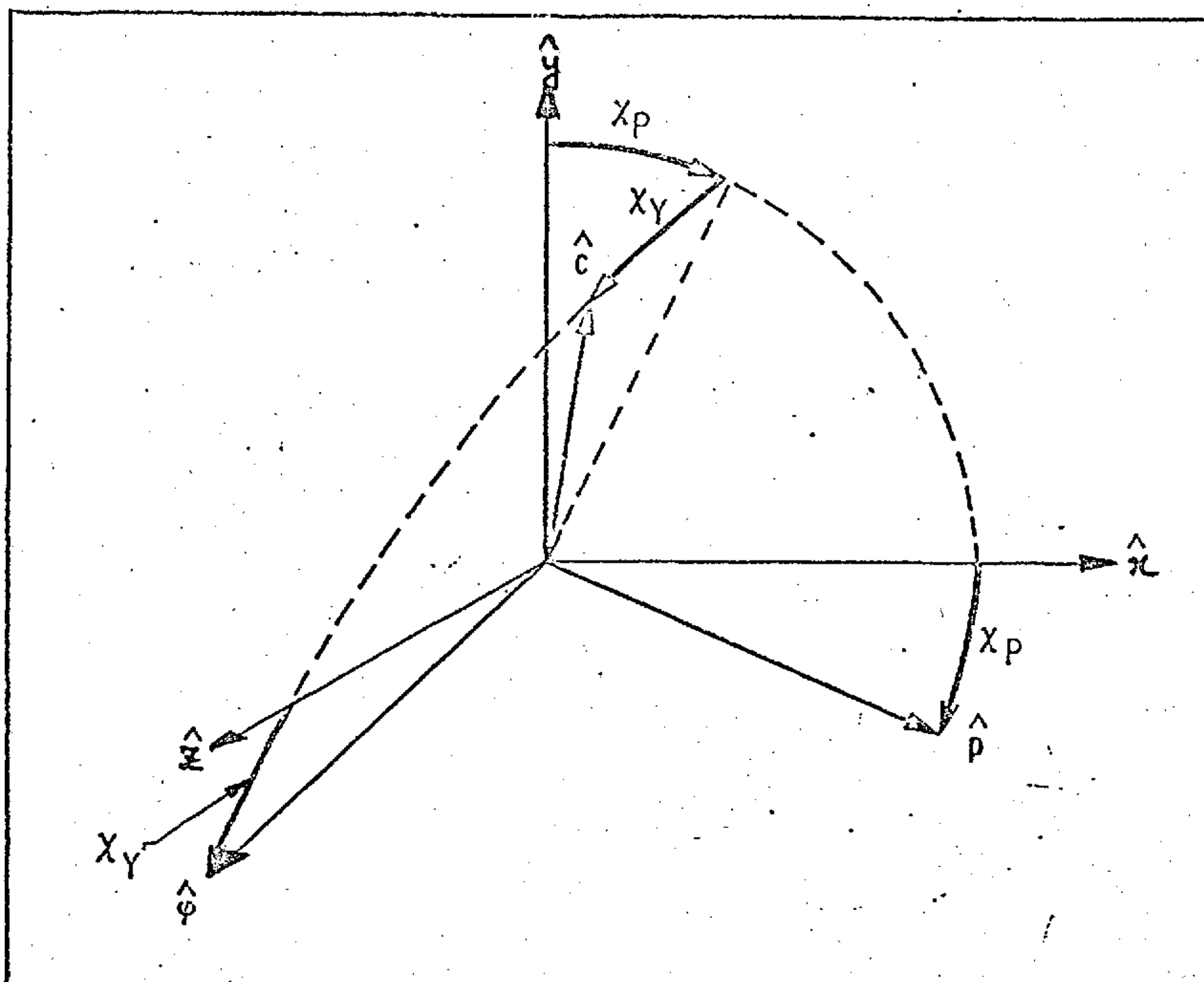


FIG. 2. χ_p, χ_Y AND $\hat{p}\hat{c}\hat{\varphi}$ COORDINATE SYSTEM

The auxiliary coordinate system is connected to the plumbline system by the transformation matrix C such that

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = C \begin{bmatrix} \hat{p} \\ \hat{c} \\ \hat{\phi} \end{bmatrix} \qquad \text{where } C = \begin{pmatrix} \cos\chi_p & \sin\chi_p \cos\chi_y & -\sin\chi_p \sin\chi_y \\ -\sin\chi_p & \cos\chi_p \cos\chi_y & -\cos\chi_p \sin\chi_y \\ 0 & \sin\chi_y & \cos\chi_y \end{pmatrix}$$

In most control applications χ_y is kept to zero in which cases C reduces to

$$C = \begin{pmatrix} \cos\chi_p & \sin\chi_p & 0 \\ -\sin\chi_p & \cos\chi_p & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2.1.2 Geophysical Properties

The gravitational potential function $U(r, \theta)$ is in a ~~first~~ reasonable approximation

$$U(r,\theta) = \frac{\mu_e}{r}\left[1 + \frac{CJ}{3}\left(\frac{R_e}{r}\right)^2 (1 - 3\cos^2\theta)\right]$$

where CJ, R_e and μ_e are input parameters which are preset ~~by~~ JNI to

$CJ = 1.62345 \times 10^{-3}$

R_e = equatorial radius = 6378165 m = 20,925,738 feet

μ_e = product of universal gravitational constant and earth mass
$= 3.986032 \times 10^{14}\ m^3/sec^2 = 1.407656 \times 10^{16}\ feet^3/sec^2$

and r, θ refer to spherical coordinates in the plumbline system. One may refer to components of $U(r,\theta)$ with respect to the plumbline position coordinates as g_x g_y g_z respectively where

$$\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = G_{11}\begin{bmatrix} x \\ y \\ z \end{bmatrix} - G_{T0}\begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix}$$

where

$$G_{11} = -\frac{\mu_e}{r_3}\left[1 + CJ\left(\frac{R_e}{r}\right)^2 (1 - 5\cos^2\theta)\right]$$

$$G_{T0} = \frac{\mu_e}{r^2}\left[2\,CJ\left(\frac{R_e}{R}\right)^2 \cos\theta\right]$$

As a first approximation one may assume a spherical earth when

$$G_{11} = -\frac{\mu_e}{r^3}$$

$$G_{T0} = 0$$

If one does assume the earth to be an ellipsoid one needs to know the flattening constant f where

$$f = \frac{1}{298.3}$$

and the angular velocity of the earth around its axis is*

$$\Omega_e = 7.2921158 \times 10^{-5} \text{ radians/sec}.$$

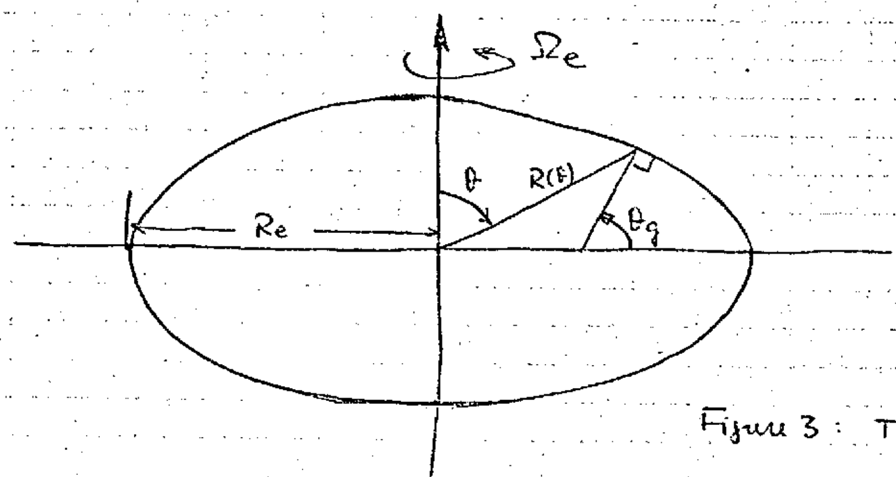


Figure 3: The earth - an ellipsoid.

The relationship between geocentric colatitude, θ, and geodetic latitude θ_g is seen to be

$$\frac{1}{\tan\theta} = (1-f)^2 \tan\theta_g$$

The radius of the earth as a function of the colatitude is

$$R(\theta) = (1-f)\, R_e \Big/ \sqrt{(1-f)^2 \sin^2\theta + \cos^2\theta}$$

2.1.3 Atmospheric Properties

Since the PRAG3 model atmosphere routine of the MSFC System was not available to the author, the following assumptions were made:

(i) atmospheric data may be looked up from a table with sufficient accuracy for exploratory calculations

(ii) the following table from the U.S. Air Force Air Research and Development Command (ARDC) model atmosphere was used by the routine NEWDRAG

~~(iii)~~

(iii) any other atmospheric table can be easily substituted.

TABLE 1. ARDC model atmosphere.

	Region	h' (feet)	$\theta \equiv T/T_0$	$\delta \equiv p/p_0$	$\sigma \equiv \rho/\rho_0$	μ/μ_0
Sea level		0	1	1	1	1
	Troposphere	5,000	0.9656	8.320×10^{-1}	8.617×10^{-1}	0.9731
	Troposphere	10,000	0.9312	6.877×10^{-1}	7.385×10^{-1}	0.9457
	Troposphere	15,000	0.8969	5.643×10^{-1}	6.292×10^{-1}	0.9178
	Troposphere	20,000	0.8625	4.595×10^{-1}	5.328×10^{-1}	0.8894
	Troposphere	25,000	0.8281	3.711×10^{-1}	4.481×10^{-1}	0.8605
	Troposphere	30,000	0.7937	2.970×10^{-1}	3.741×10^{-1}	0.8311
	Troposphere	35,000	0.7594	2.353×10^{-1}	3.099×10^{-1}	0.8011
Tropopause		36,089	0.7519	2.234×10^{-1}	2.971×10^{-1}	0.7945
	Stratosphere	40,000	0.7519	1.851×10^{-1}	2.462×10^{-1}	0.7945
	Stratosphere	45,000	0.7519	1.455×10^{-1}	1.936×10^{-1}	0.7945
	Stratosphere	50,000	0.7519	1.145×10^{-1}	1.552×10^{-1}	0.7945
	Stratosphere	60,000	0.7519	7.078×10^{-2}	9.414×10^{-2}	0.7945
	Stratosphere	70,000	0.7519	4.377×10^{-2}	5.821×10^{-2}	0.7945
	Stratosphere	80,000	0.7519	2.707×10^{-2}	3.600×10^{-2}	0.7945
Stratopause		82,021	0.7519	2.456×10^{-2}	3.267×10^{-2}	0.7945
	Mesosphere	90,000	0.7772	1.684×10^{-2}	2.167×10^{-2}	0.8167
	Mesosphere	100,000	0.8089	1.068×10^{-2}	1.320	0.8442
	Mesosphere	150,000	0.9676	1.389×10^{-2}	1.436×10^{-3}	0.9746
	Mesosphere	154,199	0.9809	1.189×10^{-3}	1.212×10^{-3}	0.9851
	Mesosphere	173,885	0.9809	5.756×10^{-4}	5.868×10^{-4}	0.9851
	Mesosphere	200,000	0.8566	2.058×10^{-4}	2.402×10^{-4}	0.8845
	Mesosphere	250,000	0.6186	1.738×10^{-5}	2.810×10^{-5}	0.6718
	Mesosphere	259,186	0.5749	9.964×10^{-6}	1.733×10^{-6}	0.6293
		295,276	0.5749	1.031×10^{-6}	1.793×10^{-6}	0.6293

where: h = altitude in feet;
T = absolute temperature;
p = absolute pressure;
ρ = density;
μ = viscosity;
g = acceleration due to gravity,

Subscript $(\ )_0$ = value at sea level;

$$h' = \frac{1}{g_0}\int_0^h g\,dh, \text{ geopotential altitude in feet;}$$

$g_0 = 32.174$ ft/sec^2;
$T_0 = 518.69°R$;
$p_0 = 2116.2$ lb$_f$/ft^2;
$\rho_0 = 0.0023769$ slug/ft^3;
$\mu_0 = 3.7373 \times 10^{-7}$ lb$_f$ sec/ft^2.

For the lower regions of this model atmosphere the geopotential is virtually equal to the physical altitude; the region up to 65,000 ft is practically identical with the International Civil Aeronautics Organization (ICAO) atmosphere.

Note that $P_0 = 2116.2$ lbs/ft^2 differs from the NASA standard of 2117.0604 lbs/ft^2,

2.1.4. Aerodynamic Forces

The passage of the vehicle through the atmosphere creates aerodynamic forces defined to act in the direction of and normal to the vehicle body axis $\hat{c}$.

The velocity V_R (VELREL) is the velocity of the vehicle relative to the atmosphere. The orientation of the $\vec{V}_R$ vector in the auxiliary coordinate system is defined by the angle of attack α and the relative velocity heading angle σ (SIGMA). The aerodynamic axial force is called F_{AA} (FAA) and the aerodynamic normal force is called F_{AN} (FAN). The above quantities are shown in Figure 3

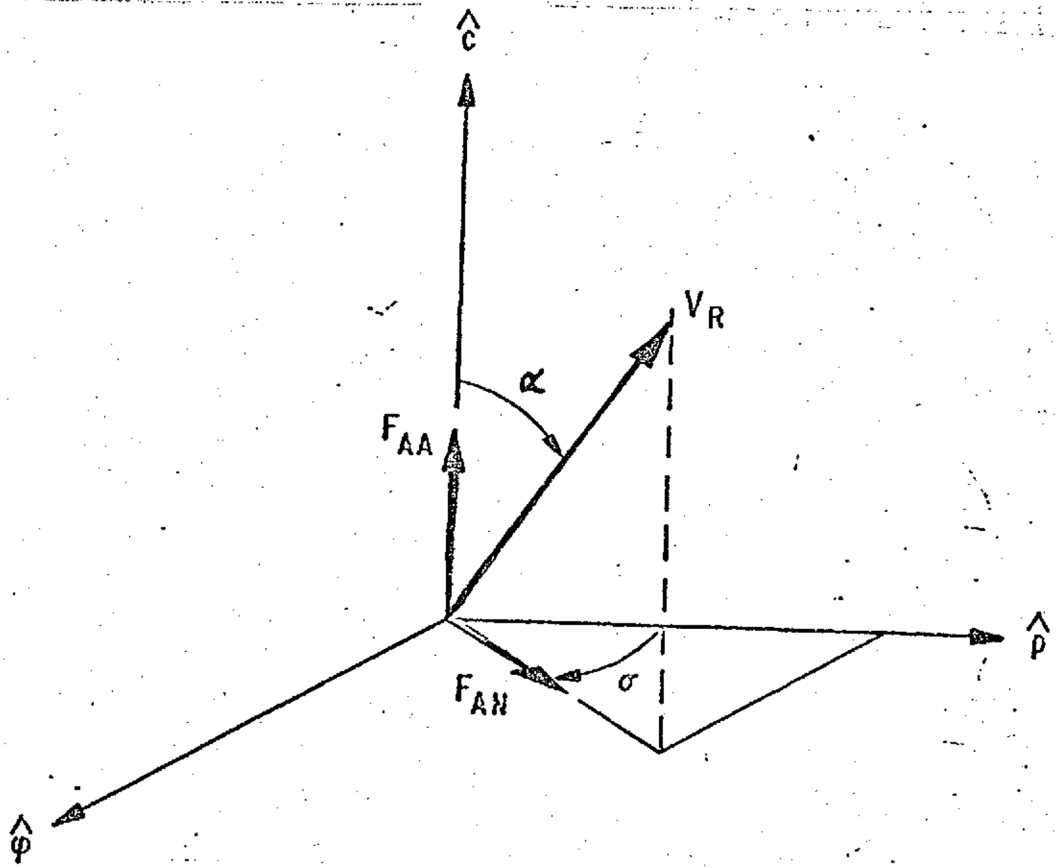


FIG. 3. AERODYNAMIC FORCE QUANTITIES

The equations for transforming V_R into the plumbline system are

$$\begin{bmatrix} \underline{W} \\ \underline{u} \\ \underline{V} \end{bmatrix} = V_R \, C \begin{bmatrix} \sin\alpha \cos\sigma \\ \cos\alpha \\ \sin\alpha \sin\sigma \end{bmatrix}$$

[illegible]

defined in 2.1.1 above.

One can also calculate the relative velocity as the difference between the inertial velocity (plumbline system) and the transformed velocity required by an object in order to remain over a given point on the rotating earth. This gives

$$\begin{bmatrix} \underline{w} \\ \underline{u} \\ \underline{v} \end{bmatrix} = \begin{bmatrix} w - (a_{22} z - a_{32} y)\,\Omega_e \\ u - (a_{32} x - a_{12} z)\,\Omega_e \\ v - (a_{12} y - a_{22} x)\,\Omega_e \end{bmatrix}$$

where $(\underline{w}, \underline{u}, \underline{v})$ are the relative velocity components in the plumbline system and (w, u, v) are the inertial velocity components in the plumbline system in the directions $\hat{x}, \hat{y}, \hat{z}$ respectively. We have

$$V_R = \sqrt{\underline{w}^2 + \underline{u}^2 + \underline{v}^2}$$

$$\alpha = \arccos\left(\frac{\underline{w}}{V_R}\sin\chi_p\cos\chi_y + \frac{\underline{u}}{V_R}\cos\chi_p\cos\chi_y + \frac{\underline{v}}{V_R}\sin\chi_y\right)$$

$$\sigma = \arctan\left(\frac{-\underline{w}\sin\chi_p\sin\chi_y - \underline{u}\cos\chi_p\sin\chi_y + \underline{v}\cos\chi_y}{\underline{w}\cos\chi_p - \underline{u}\sin\chi_p}\right)$$

which may be reduced to the following simpler expressions if $\chi_y = 0$

$$\alpha = \arccos\left(\frac{\underline{w}}{V_R}\sin\chi_p + \frac{\underline{u}}{V_R}\cos\chi_p\right)$$

$$\sigma = \arctan\left(\frac{\underline{v}}{\underline{w}\cos\chi_p - \underline{u}\sin\chi_p}\right)$$

The dynamic pressure q (Q) is calculated as

$$q = \frac{1}{2}\rho V_R^2$$

where ρ, the density is looked up in the standard atmosphere table. The Mach number may be calculated as

$$M = \frac{V_R}{a}$$

where a is the speed of sound and Mach number may be used the drag coefficient C_A. Alternately one may input C_A from a table where C_A is tabulated against the Mach number.

The axial force may now be calculated as

$$F_{AA} = q\ S\ C_A$$

where S is the reference area of the vehicle and is an input constant. Similarly the normal force may be calculated as

$$F_{AN} = q \cdot S \cdot C_N' \cdot \alpha$$

where $C_N'(M)$ is again tabulated and provided as a table to be looked up with respect to the Mach number.

The aerodynamic forces in the $\hat{x}\ \hat{y}\ \hat{z}$ directions are calculated by transforming the simple $\hat{p}\ \hat{c}\ \hat{\phi}$ system forces to the plumbline system using the matrix C defined in 2.1.1.

$$\begin{bmatrix} F_{Ax} \\ F_{Ay} \\ F_{Az} \end{bmatrix} = -C \begin{bmatrix} F_{AN}\cos\tau \\ F_{AA} \\ F_{AN}\sin\tau \end{bmatrix}$$

The minus sign is used because the velocity of the atmosphere relative to the vehicle is the negative of $\vec{V_R}$

2.1.5. Booster Configuration

This system assumes that certain characteristics are common to each stage of a multistage vehicle. These quantities are initialized by the routine INSTAG1 and the user can provide and invoke any number of subsequent stages by writing similar routines. Each stage is divided into five phases mainly recognized by the different treatment of the control variable χ_p.

In this sense any phase of any stage is a "thrust-event" whose duration may be selected by the user of the system. Each thrust event is characterized by the following quantities:

THRENG	lbs	thrust per engine
MDOT	lbs/sec	mass flow per engine
NENG		number of engines
TPHASE (i)	sec	duration of this phase
AREANOZ	ft^2	nozzle area for one engine
AREAREF	ft^2	reference area for drag calculations

For the whole stage we also need

MLBS	lbs	mass of the vehicle including fuel and payload
MFUEL	lbs	mass of propellant in the stage

If the thrust is given as nominal sea level thrust, the total thrust is given by

$$THRUST = NENG * \left(THRENG_{s.l.} + AREANOZ\,(P_s - P_a) \right)$$

where P_s and P_a are sea-level and current pressures respectively. For vacuum thrust we calculate the total thrust as,

while the vehicle is in the atmosphere

$$THRUST = NENG * (THRENG_{vac.} - AREANOZ * p_a)$$

and outside the atmosphere as

$$THRUST = NENG * THRENG$$

where p_a is the current pressure which becomes zero outside the atmosphere. The sea-level and vacuum thrust expressions are therefore connected by

$$Th_{vac} = Th_{sea.lev.} + AREANOZ * P_s$$

$$= Th_{sea.lev.} + 2116.2 * AREANOZ.$$

2.1.6. The Equations of Motion

The equations of motion are

$$\begin{bmatrix} \dot{w} \\ \dot{u} \\ \dot{v} \end{bmatrix} = C \begin{bmatrix} -F_{AN} \cos \nabla \\ T - F_{AA} \\ -F_{AN} \sin \nabla \end{bmatrix} g_0/m \quad + \quad G_{11} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} w \\ u \\ v \end{bmatrix}$$

Since $\dot{m}$ is constant or a function of time only when vehicle is throttled the mass flow equation can be integrated analytically and at each step

$$dm = \dot{m}\, dt$$

The solution integrates the above six equations of motion using the standard fourth order Runge - Kutta procedure and at each step updates the total mass as needed to take account of propellant burned during the step.

Thus weight drops may be produced by the user at any stopping point by simply subtracting the dropped mass from the total mass

MLBS = MLBS − MDROP

All the quantities in the above equations of motion have been defined previously except

g_0 = 32.1740486 ft/sec²

m = (MLBS) mass of vehicle in lbs including payload

2.1.7. The Initial Conditions

If the geodetic co-latitude is θ_0 then

$$\theta_1 = \pi/2 - \theta_0 \quad \text{for a spherical earth}$$

$$\theta_1 = \pi/2 - \tan^{-1}\left((1-f)^2 \tan\theta_0\right) \quad \text{for an earth with flattening } f \text{ where } f = \frac{1}{297.3}$$

The radius of the earth is

$$R_e = 20{,}925{,}738 \text{ feet} \quad \text{for a spherical earth}$$

$$R(\theta) = (1-f)R_e \Big/ \sqrt{(1-f)\sin^2\theta + \cos^2\theta} \quad \text{for a non-spherical earth.}$$

The initial velocities in the plumbline system are

$$\begin{bmatrix} w_0 \\ u_0 \\ v_0 \end{bmatrix} = V_0 A \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} V_0 \sin(AZIM) \\ 0 \\ V_0 \cos(AZIM) \end{bmatrix}$$

where AZIM is the azimuth angle of the launch site and $V_0 = R(\theta)\,\Omega_e$ where $\Omega_e = 7.292115 \times 10^{-5}$ radians/second, so that for a spherical earth

$$\begin{bmatrix} w_0 \\ u_0 \\ v_0 \end{bmatrix} = \begin{bmatrix} R_e \Omega_e \sin\theta_1 \sin(AZIM) \\ 0 \\ R_e \Omega_e \sin\theta_1 \cos(AZIM) \end{bmatrix}$$

The initial position of the vehicle at launch, assuming a vertical launch tower is

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = R(\theta)\, A \begin{bmatrix} 0 \\ \cos(\theta) \\ \sin(\theta) \end{bmatrix}$$

which for a spherical earth is

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = R_e \begin{bmatrix} a_{12}\cos\theta + a_{13}\sin\theta \\ 1 \end{bmatrix} = R_e \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ R_e \end{bmatrix}$$

2.1.8. Phases of the Flight.

The system is programmed to automatically enter the next phase at prescribed times and to save the state of each beginning of a phase so that a user can at any time return to the beginning of any previously entered phase and reintegrate the equations of motion under changed conditions.

The phase sequence may be easily changed by rewriting the routine NEWCHI. The user is encouraged to do this as our initial choice may not be optimal for his current application. The initial choice, currently programmed, is:

Phase 1: vertical rise characterized by

$$\chi_p = 0$$
$$\chi_y = 0$$

Phase 2: pitch-over phase characterized by

$$\chi_p = \dot{\chi}\,(T - T_L)$$
$$\chi_y = 0$$

where $\dot{\chi}$ (CHIDOT) is an input constant equal to the rate of change of χ_p (CHIP) and T_L is the beginning of the pitch-over phase calculated and set by the system from user-input phase duration times

Phase 3: gravity turn phase characterized by

$$\chi_p = \arctan\left(\frac{w}{u}\right)$$
$$\chi_y = 0$$

Phase 4: Chi-freeze phase characterized by

$$\chi_p = \text{constant}$$
$$\chi_y = \text{constant}$$

Phase 5 control turn phase characterized by

$$\chi_p = a_1 + a_2\,(T - T_{EL})$$
$$\chi_y = 0$$

where a_1 is the value of χ_p upon entry into phase 5 and a_2 is a user supplied [illegible]

is as a phase 2 supplied by the system from user input phase duration times.

One notes that in all these manoeuvers one tries to keep $X_y = 0$ hence to emphasize the design of the interactive program system we have restricted ourselves to the case $X_y = 0$ in the actual programs. The full X_y dependent forms may be recovered by a trivial clerical task of updating all appropriate expression codes using the expressions given in this section.

2.2 Interactive Requirements.

To perform interactive trajectory shaping and optimization a user needs:

(i) to fly a vehicle with one or more stages, each with a different number of engines, thrust, burn-rate and so on.

(ii) during the flight of each stage perform different control manoeuvers. These are called phases or ~~[illegible]~~ simulation sections.

It seems desirable to give the user the ability to step back to the start of any previous simulation section and restart the flight there with different parameters.

(iii) to stop at any time in the flight with full control over all defined variables or programs so that one can examine progress to date and either continue or change some parameters and continue or step back to a previous simulation section, change some parameters and continue.

(iv) to be able to display the trajectory graphically and numerically

(v) to stop the flight whenever a user set constraint is violated. Typical constraints are maximum g-force, maximum dynamic pressure or maximum altitude.

(vi) to drop a mass (for example the exhausted first stage may be dropped) at any time in the flight

(vii) to throttle automatically if G exceeds a prescribed GMAX to hold G at GMAX

(viii) to shut off one or more engines

(3.0) THE SOLUTION

In order to be useful, the interactive system must be flexible, modular, and transparent to the user.

Flexibility means that the user should have maximum control at any point at which he desires to stop the flight. This avoids elaborate control point logic which is in any case unsatisfactory for most users. It makes every point into a control point. The user selects TSTOP = t and the time t becomes a control point at which the user can

(i) examine the trajectory
(ii) change any parameter
(iii) change any routine
(iv) continue the ~~trajectory~~ flight
(v) step back to the beginning of any previous phase and do (i) (ii) (iii) (iv)
(vi) abandon the flight.

Flexibility is achieved by a coherent naming convention and use of macros or in Fortran terminology by putting all variables into named COMMON. They are thereby available at any stopping point whatever.

Modularity means that each routine is designed to perform a specific function and only this function so that only one place in the set of routines is affected if the user decides to change forces or use different control techniques in a phase or calculate atmospheric drag differently or reduce the number of engines and so on.

Transparency means that the user should be able to see at a glance whether a routine corresponds to its mathematical formulation or not. To achieve this the names are all chosen to be mnemonics of the corresponding mathematical names and the expressions are written so that they preserve their similarity to the mathematical formulas of section two.

One may argue that this approach prevents the ultimate optimization of the computation sequence on the computer but since the ratio between the execution cost of a computer instruction and the writing cost of an instruction is between 10^{-6} and 10^{-9}, it seems more important that the user can be sure that his first flight gives the trajectory that he had in mind so that he can concentrate on his task of trajectory optimisation ~~and not~~

without being continuously diverted by the need to verify that the trajectory output by the computer is actually the trajectory which the user desires and is not corrupted by some bug in the optimization of expression evaluation.

3.1 The Language

The language chosen for the system was SIGMA [3] for the following reasons:

(i) SIGMA provides the most powerful general purpose interactive graphics capability anywhere

(ii) SIGMA is sufficiently AMTRAN-like to allow reasonably easy translation into AMTRAN

(iii) SIGMA is sufficiently FORTRAN-like to allow translation into FORTRAN. All variables then go into named common and the MACRO's become Fortran subroutines. Automatic array handling is purposely kept to a minimum to facilitate translation and where it occurs it can be done by an axiliary service routine written in FORTRAN

(iv) SIGMA allows statement by statement execution and hence gives the user interactive control over his program execution.

(v) SIGMA graphics is available on storage type CRT's as required by the scope of work.

SIGMA, is a direct descendant of the AMTRAN language and system (started by Dr R. N. Seitz at MSFC in ~~early~~ late 1964) and hence it incorporates many of AMTRAN's features. SIGMA is under development at CERN, Geneva Switzerland, and University of Georgia, Athens Georgia on the CDC 6000 series machines and at the University of Saskatchewan, in Saskatoon Canada on the IBM 360/370 series machines.

3.2 The Layout of the Solution

This section describes the general design of the solution. All variables are named as close to their mathematical names as possible. All names are global throughout the system, for example T always denotes time. An alphabetical list of names is included in Appendix A.

It is often desirable to treat the state variables (w, u, v, x, y, z) as a vector. Since SIGMA does not have the concept of equivalence, a vector of seven components STATE = (w, u, v, x, y, z, t) was defined and a routine SETVAR is used to perform "the equivalence function" to set W = STATE(1) and so on.

The only variables whose function is not obvious from their name are

w	=	STATE (1)	⟷	NSTATE (1)	⟷	W
u	=	STATE (2)	⟷	NSTATE (2)	⟷	U
v	=	STATE (3)	⟷	NSTATE (3)	⟷	V
x	=	STATE (4)	⟷	NSTATE (4)	⟷	X
y	=	STATE (5)	⟷	NSTATE (5)	⟷	Y
z	=	STATE (6)	⟷	NSTATE (6)	⟶	Z
t	=	STATE (7)	⟷	NSTATE (7)	⟶	T

The auxiliary vector NSTATE is necessary because the Runge-Kutta scheme makes three partial steps before taking a full step (see RUNGE).

The state of the beginning of each phase is stored in a two dimensional 5 × 11 component array named PHASE. Each row of Phase stores for the corresponding phase the following items (w u v x y z t total mass fuel mass X_p and thrust).

Mass is needed because MCBS contains the actual mass of the vehicle at any instant which has to be reset on steps back.

Fuel mass is needed to warn the user that vehicle has run out of fuel and to provide automatic coasting

X_p is computed so that a step back does not automatically recover the old value of X_p in all phases

THRUST is a function of time if engines are throttled to keep G below GMAX.

The vector TRAJ saves any or all items specified in the Routine SAVTRAJ for each point at which SAVTRAJ is called. Since phase boundaries are also saved in TRAJ, the number of items saved is stored in the variable TRAJITH.

There is only one logical control variable INPHASE which contains the phase number of the currently executing phase. It may take the values $1 \leq$ INPHASE ≤ 5 because array PHASE and routine NEWCHI are not programmed to cope with more. Any extensions to this are therefore obvious and simple.

The duration of each phase is stored in a vector TPHASE and the user has to preset TPHASE$(1 \rightarrow 5)$ to any set of desired times.

All other variable names have meanings which are easily recognizable from the names of the corresponding quantities in section 2.

An alphabetical list of all routines follows:

CA(ALT)	function of altitude	looks up tables and calculates C_A for drag F_{AA}
CNP(ALT)	function of altitude	looks up tables and calculates C_N' for drag F_{AN}
DENSITY(ALT)	function of altitude	looks up table to get atmospheric density
FLY	macro	main routine to advance trajectory
INI	macro	initializes all variables
INSTAG1	macro	initializes all variables typical of a rocket stage
MACH(ALT)	function	calculates or looks up Mach number
NEWCHI	macro	calculates a new value of the quantities named: x_p
NEWDRAG	macro	calculates a new value of the quantities named: F_{AA} F_{AN}
NEWRHS	macro	calculates a new value of the quantities named: evaluates right-hand sides of state eqns.
NEWTHRS	macro	calculates a new value of the quantities named: THRUST
NOFUEL	macro	allows the vehicle to coast without thrust.
RUNGE	macro	integrates equations of state over one step H in time T using fourth-order Runge-Kutta integration
SAVTRAJ	macro	saves a specified sequence of quantities by concatenating it to the end of the trajectory vector TRAJ.
SETVAR	macro	sets all variables which depend upon the state vector
TRAJSHO	macro	displays the trajectory vector in some convenient way.

3.3 Logical Structure of the Main Routines

INI initializes all variables for a fresh launch

- calls → INSTAG1 — initializes all variables associated with first stage
- calls → SETVAR — sets or calculates individually named variables
  - calls → NEWCHI — calculates new χ_p
  - calls → NEWDRAG — calculates new F_{AA} F_{AN}
  - calls → NEWTHRS — if throttled calculates new THRUST
- Calls → SAVPHAS — produces first entry in PHASE array to save starting point of first phase.

FLY loops TSTOP times

- calls → RUNGE — calculates state at $t = t + h$
  - calls → NEWRHS — evaluates state equation right-hand sides
  - loop 3 times
    - calls → SETVAR
      - calls → NEWCHI
      - calls → NEWDRAG
      - calls → NEWTHRS
    - calls → NEWRHS
  - sets → STATE — sets new state vector
  - Calls → SETVAR — distributes state vector to individually named variables.
    - calls → NEWCHI
    - calls → NEWDRAG
    - Calls → NEWTHRS
- calls → SAVETRAJ
- calls → TRAJSHO

REPHAS uses array PHASE to restore all individually named variables from the row PHASE(INPHASE,) where INPHASE is a user selected previously entered phase.

- calls → **SETVAR**
  - calls → **NEWCHI**
  - calls → **NEWDRAG**
  - calls → **NEWTHRS**
- calls → **SAVTRAJ** makes a mark in the trajectory vector that a return to a previous phase took place

SAVPHAS advances INPHASE by one and saves the present state in a row PHASE(INPHASE,) of the ~~vector~~ array PHASE

- calls → **SAVTRAJ** makes a mark in the trajectory vector to indicate which phase is now entered.

The logical structure of all other routines is trivial.

3.4 A Brief Description of each Routine

This section gives the source code of each routine together with some comments on its purpose and on SIGMA language features which may not be familiar to the reader.

3.4.1 FLY

```
44.  $-----------------------------------------------------------
45.  MACRO FLY
 2.  $-----------------------------------------------------------
 3.  $    ADVANCES TRAJECTORY EITHER BY 200 STEPS OR TILL ONE OF
 4.  $    THE USER SPECIFIED LIMITS IS EXCEEDED
 5.  $
 6.  DO 10 I=1,200                                          Do loop like FORTRAN
 7.  $    ENTER A NEW PHASE AUTOMATICALLY WHEN NECESSARY
 8.  IF(T LT TPHSTOP) GOTO 5                                IF like FORTRAN
 9.  $    AT OR OVER PHASE STOP,SAVPHAS, THEN SET ALL VAR. THAT CHANGE
10.  CALL SAVPHAS
11.  CALL NEWCHI
12.  5  CONTINUE
13.  IF(T GE TSTOP) GOTO 100
14.  IF(Q GT QMAX) GOTO 100
15.  IF(ALT GT ALTMAX) GOTO 100
16.  IF(MFUEL LE 0) CALL NOFUEL
17.  $
18.  CALL RUNGE
19.  $                                           MOD is the remaindering
20.  NOWSTEP=NOWSTEP+1                           function of FORTRAN
21.  IF(MOD(NOWSTEP,SAVSTEP) EQ 0) CALL SAVTRAJ  if NOWSTEP is a multiple of SAVSTEP
22.  10 CONTINUE
23.  $
24.  $
25.  PRINT'YOU HAVE COMPLETED 200 STEPS WITHOUT REACHING ANY LIMIT'
26.  $
27.  100 CONTINUE
28.  PRINT T,TSTOP,G,GMAX,Q,QMAX,ALT,ALTMAX,MFUEL
29.  CALL TRAJSHO
30.  END
46.  $
```

FLY is the routine which advances the trajectory.

6. The DO-loop safeguards against accidental endless runs – as for example TSTOP accidentally is set negative or whatever other reason. It also limits the size of TRAJ on first run – important if the system is translated to FORTRAN where TRAJ would have to be dimensioned to some fixed size.

8. enters a new phase automatically by calling SAVPHAS and NEWCHI when ever T reaches TPHSTOP. TPHSTOP is set for each next phase by SAVPHAS.

13, 14, 15 user selected limits which stop the integration and generate an automatic decision break point when one of these limits is reached.

18 integrates to next point

20, 21, 22 decides if NOWSTEP is an exact multiple of SAVSTEP and if it is saves this point in TRAJ by calling SAVTRAJ.

25 Indicates to the user that break point is not from any of his IF-test limits but because he has stepped 200 steps in time.

It seemed desirable to include this "outer limit" in addition to the TSTOP limit because on long trajectories with small step sizes the user may be unaware that something may be wrong so a glance at the trajectory at each 200th step was included. To continue the user has to simply CALL FLY.

27 At the conclusion prints the values of all IF-test items and their limits for two reasons

(i) to show the user which limit was reached at this break point

(ii) to remind the user that all these quantities are tested at each step so that he can remove obsolete limits or tests whenever they are no longer needed.

29 shows the trajectory either on printer or graphic display

3.4.2. RUNGE


```
$-----------------------------------------------------------------
MACRO RUNGE
$-----------------------------------------------------------------
S    ON ENTRY STATE AND IDIVIDUAL VARIABLES SHOULD MATCH
$    DURINT RUNGE STATE IS UNTOUCHED TILL THE END
$    AT END STATE IS UPDATED TOGETHER WITH THE INDIVIDUAL VARIABLES
$    NOTE THAT MLBS AND WHEN THROTTLED MDOT THRUST ARE FNS OF TIME
$
CALL NEWRHS
K1=H*RHS
$
NSTATE=STATE+K1/2
CALL SETVAR
MLBS=MLBS-NENG*MDOT*H/2
CALL NEWRHS
K2=H*RHS
$
NSTATE=STATE+K2/2
CALL SETVAR
CALL NEWRHS
K3=H*RHS
$
NSTATE=STATE+K3
CALL SETVAR
MLBS=MLBS-NENG*MDOT*H/2
CALL NEWRHS
K4=H*RHS
$
$
STATE=K1/6+K2/3+K3/3+K4/6+STATE
NSTATE=STATE
CALL SETVAR
MFUEL=MFUEL-NENG*MDOT*H
G=(THRUST-FAA)/MLBS
END
$
```

In this routine SIGMA notation is identical with FORTRAN except that $ ⇒ C for comments.

RUNGE uses fourth order Runge-Kutta formula to generate the dependent variable values at T + H when they are known at T

On entry the routine assumes that all individually named variables are properly set and does not call SETVAR. Variables may be set in

(i) INI

(ii) REPHAS

(iii) just before exit from RUNGE on the previous step.

RHS is a vector of 7 items generated by the routine NEWRHS. The first six items are the right-hand-sides of the six equations of state and the seventh is equal to one. When multiplied by H, the seventh

The mass equation is not integrated numerically since mass is a function of time only. Instead, at each step mass is updated as follows:

(i) whenever time changes by H/2, such as at statement 13 and 24, mass is updated using MDOT the mass flow per engine and NENG, the number of engines. In this form the update will be correct regardless of the number of engines.

Just before exit from this routine, mass of fuel MFUEL and G-force are recalculated. Also all variables are set so that they are

(i) ready for the next entry into RUNGE

(ii) correct and available if FLY decides that this is a breakpoint and returns control to the user.

3.4.3 NEWRHS

```
40.  $-------------------------------------------------------------
41.  MACRO NEWRHS
 2.  $-------------------------------------------------------------
 3.  COSCHIP=COS(CHIP)
 4.  SINCHIP=SIN(CHIP)
 5.  SIGMA=ATAN2(VREL,WREL*COSCHIP-UREL*SINCHIP)
 6.  GRAVIT=MUGRAV/((X*X+Y*Y+Z*Z)**1.5)
 7.  DW=(-COSCHIP*FAN*COS(SIGMA)+SINCHIP*(THRUST-FAA))/MLBS*GZERO-X*GRAVIT
 8.  DU=(SINCHIP*FAN*COS(SIGMA)+COSCHIP*(THRUST-FAA))/MLBS*GZERO-Y*GRAVIT
 9.  DV=-FAN*SIN(SIGMA)/MLBS*GZERO-Z*GRAVIT
10.  $
11.  RHS=DW&DU&DV&W&U&V&1
12.  END
```

NEWRHS calculates the right-hand sides of the equations of motion

3, 4 just to save time

5 SIGMA is the angle σ of section 2, ATAN2 is the FORTRAN arctan routine

6 μ / r^3 computed just once

7, 8, 9 first three equations

The right hand sides of the next three equations are W U V resp.

11 the right-hand sides are returned as a vector using the concatenation operator of the SIGMA language (&).

C = A & B means take the vector A join B head-first to the tail-end of A and call this new vector C.

In FORTRAN statement 11 would be replaced by direct assignment to the components of RHS or by equivalencing DW DU DV etc to the components RHS(1) RHS(2) RHS(3) etc.

3.4.4. SETVAR

```
$----------------------------------------------------------------
MACRO SETVAR
$----------------------------------------------------------------
$   SET OR CALCULATE ALL INDIVIDULLY NAMED VARIABLES
$   EXPECTS THE CURRENT STATE VECTOR IN  NSTATE
$
W=NSTATE(1)
U=NSTATE(2)
V=NSTATE(3)
X=NSTATE(4)
Y=NSTATE(5)
Z=NSTATE(6)
T=NSTATE(7)
$
$
WREL=W-OMEGA*(A22*Z-A32*Y)
UREL=U-OMEGA*(A32*X-A12*Z)
VREL=V-OMEGA*(A12*Y-A22*X)
$
VELREL=SQRT(WREL*WREL+UREL*UREL+VREL*VREL)
$
$
CALL NEWCHI
CALL NEWDRAG
CALL NEWTHRS
$
END
```

SETVAR recalculates all variables which change at each step of the integration.
The user may include more items or exclude some as his needs change.

3.4.5 NEWTHRS

```
$----------------------------------------------------------------
MACRO NEWTHRS
$----------------------------------------------------------------
IF(G LT GMAX) RETURN:
$   REDUCE MDOT AND RECALCULATE THROTTLED THRUST
THRUST=GMAX*MLBS+FAA
MDOT=THRUST/ISP
END
$
$
```

NEWTHRS leaves THRUST as set by INI or REPHAS and recalculates it only when throttling a stage to hold a maximum G-force.

If it seems desirable to apply the atmospheric thrust correction this routine should be modified as follows

statement 3 should be replaced by the following statements.

```
      IF (G GE GMAX) GOTO 99
   $ calculate unthrottled thrust (if THRENG is given as nominal sea-level thrust/engine)
      THRUST = NENG * (THRENG + AREANOZ * (1 - PRESS(ALT))/2117.0624)
      RETURN
   99 CONTINUE
```

and where PRESS(ALT) is a call of the function PRESS which looks up the pressure in a table calculated from the standard atmosphere.

All these table look-up functions are trivial in Sigma and almost as simple in FORTRAN so only one sample is given for DENSITY(ALT) and others are left to the user to choose the best table length for his particular application.

3.4.6 NEWCHI

```
$-----------------------------------------------------------------
MACRO NEWCHI
$-----------------------------------------------------------------
IF(INPHASE NE 0) GOTO 1
$    FIRST INITIALIZATION
CHIP=0
CHIY=0
RETURN
$
1  CONTINUE
IF(INPHASE NE 1) GOTO 2
$    PHASE  ONE
$    VERTICAL RISE
CHIP=0
CHIY=0
RETURN
$
2  CONTINUE
IF(INPHASE NE 2) GOTO 3
$    PHASE  TWO
$    TILT-OVER, STARTING TIME OF PHASE IN PHASE(INPHASE,7)
CHIP=CHIDOT*(T-PHASE(INPHASE,7))
CHIY=0
RETURN
$
3  CONTINUE
IF(INPHASE NE 3) GOTO 4
$    PHASE  THREE
$    GRAVITY TURN
CHIP=ATAN2(WREL,UREL)
CHIY=0
RETURN
$
4  CONTINUE
IF(INPHASE NE 4) GOTO 5
$    PHASE  FOUR
$    CHI-FREEZE LEAVES CHIP AND CHIY AS THEY ARE
RETURN
$
5  CONTINUE
IF(INPHASE NE 5) PRINT'INPHASE INCORRECT'
$    PHASE  FIVE
$    CONTROL TURN
CHIP=A1+A2*(T-PHASE(INPHASE,7))
CHIY=0
$
END
```

NEWCHI calculates χ_p and χ_y differently for each phase. The Sigma language does not have a computed GOTO, hence the chain of IF-statements. Any other quantity which is computed differently for each phase may be calculated with a similar routine. One has to take care however to save the quantity in phase (PHASE) if it is ~~upda~~ changed in such a way that needs a reset when stepping back to the beginning of

3.4.7. NEWDRAG

```
14.  $--------------------------------------------------------------
15.  MACRO NEWDRAG
 2.  $--------------------------------------------------------------
 3.  $
 4.  $   CALCULATES THE DRAG FORCES
 5.  $                           FAA  IS THE AXIAL DRAG
 6.  $                           FAN   IS THE NORMAL DRAG
 7.  $
 8.  Q=0.5*DENSITY(ALT)*(W*W+U*U+V*V)
 9.  FAA=Q*AREAREF*CA(MACH(ALT))
10.  FAN=0
11.  END
```

NEWDRAG calculates first the dynamic pressure Q and then the axial and normal drag forces F_{AA} and F_{AN} respectively.

DENSITY is a function for calculating or looking up the density of the atmosphere. A table look up function for density is given below.

CA is a function of the Mach number for looking up the drag-coefficient table.

MACH is a function which calculates the Mach number as

$$MACH = VELREL / a$$

where a is the speed of sound calculated from the model atmosphere or looked up in a table by a function A(ALT)

FAN the normal force is set to zero as it is quite small but a very simple replacement of statement 10 will change FAN to any expression the user may desire.

3.4.8. DENSITY MACH CA

```
23.  $------------------------------------------------------------
24.  FUNCTION DENSITY(ALTITUD)
 2.  $------------------------------------------------------------
 3.  $    LOOKS UP THE DENSITY IN A TABLE IN STEPS OF 10,000 FEET
 4.  $    ABOVE 200,000 FEET RETURNS ZERO AS OUTSIDE THE ATMOSPHERE
 5.  $
 6.  IF(ALTITUD LE 200000) GOTO 10
 7.  DENSITY=0
 8.  RETURN
 9.  $
10.  10    CONTINUE
11.  DENSITY=ROW(ALTITUD/10000)
12.  END
```

Here we give the table look-up functions which are quite simple and straight forward.

6. if table size is smaller than ALTMAX can reach this test will prevent the overstepping of the table end. This is especially necessary in FORTRAN where no check of array bounds is made. Altitude cannot be negative so lower bound is ok.

11. SIGMA language indexing automatically rounds the index value. In FORTRAN one would have to use INT (ALTITUD/10000 + 0.5) which can be used for rounding if the argument is positive.

Unfortunately the atmosphere model routine and the drag tables were not available to the user hence ROW is initialized to zero by INI and the routines for MACH and CA set to return zero value

```
17.  $------------------------------------------------------------
16.  FUNCTION MACH(ALTITUD)
 2.  $------------------------------------------------------------
 3.  $    HERE ONE NEEDS TO LOOK UP THE SPEED OF SOUND FROM A TABLE
 4.  $    SUCH A TABLE IS NOT AVAILABLE TO THE AUTHOR
 5.  MACH=0
 6.  END
19.  $
20.  $------------------------------------------------------------
21.  FUNCTION CA(MACHNUM)
 2.  $------------------------------------------------------------
 3.  $    AGAIN A TABLE LOOKUP ON A TABLE UNAVAILABLE TO THE AUTHOR
 4.  CA=0
 5.  $
 6.  END
```

All numerical and graphical data shown is run with the atmosphere neglected which is most efficiently done by setting FAA = 0, FAN = 0 in INI

and redefining NEWDRAG as the empty do-nothing macro MACRO NEWDRAG END.

If [illegible] have to supply the proper tables in INT and update the above routines

3.4.9 NOFUEL


```
$-----------------------------------------------------------------------
MACRO NOFUEL
$-----------------------------------------------------------------------
$
$    CORRECT MLBS FOR EXCESS FUEL SUBTRACTED IN LAST FUELED STEP
IF(MFUEL LT 0) MLBS=MLBS+MFUEL
MFUEL=0
$
$    ALLOW THE ROCKET TO COAST
$
MDOT=0
THRUST=0
G=0
END
```

NOFUEL is called if and only if FLY detects that the vehicle has exhausted all of its fuel.

It sets $\dot{m}$ thrust and G to zero and thereby allows the rocket to coast.

One may argue that it would be sufficient to set $\dot{m}$ = thrust = G = 0 just once but the logic to decide which is the first and which are subsequent times would complicate FLY and would be executed at each step, while the coasting period in any flight is small – of the order of 10 seconds in a flight of 300–400 seconds.

It was decided to make the fuel-exhaustion check automatically because otherwise a user might overlook fuel depletion and develop unrealistic trajectories.

3. 4. 10. SAVPHAS

```
35.  $-------------------------------------------------------------------------
36.  MACRO SAVPHAS
 2.  $-------------------------------------------------------------------------
 3.  $    SAVES THE STATE OF THE SYSTEM AT TE BEGINNING OF EACH PHASE
 4.  $    SO CALL REPHAS CAN RESTART TRAJECTORY FROM THIS POINT ONWARDS
 5.  $
 6.  INPHASE=INPHASE+1
 7.  $    CONCATENATE A COLUMN OF INPHASES IN TRAJ TO INDICATE THE
 8.  $    START OF A NEW PHASE AND THE TIME OF START OF IT
 9.  SCR=INPHASE*ARRAY(TRAJITM)
10.  SCR(7)=STATE(7)
11.  TRAJ=TRAJ&SCR
12.  $    SET CONSTANT A1 FOR CONTROL TURN IN PHASE FIVE
13.  IF(INPHASE EQ 5) A1=CHIP
14.  CALL SAVTRAJ
15.  PHASE(INPHASE,ARRAY(7,1#7))=STATE
16.  PHASE(INPHASE,8)=MLBS
17.  PHASE(INPHASE,9)=MFUEL
18.  PHASE(INPHASE,10)=CHIP
19.  PHASE(INPHASE,11)=THRUST
20.  TPHSTOP=STATE(7)+TPHASE(INPHASE)
21.  END
```

SAVPHAS saves all items that need to be reset to restart the trajectory at the beginning of a phase.

9 ARRAY (TRAJITM) generates a vector of TRAJITM components all equal to 1 hence SCR contains TRAJITM components all equal to INPHASE

10 records the time of the phase start in the phase start marker vector SCR

11 concatenates (joins end to beginning) the trajectory to the phase start marker vector

13 to provide continuity of χ_p in phase five where $\chi_p = A1 + A2 * \text{time in phase}$ we set A1 to the entrance value of χ_p.

20 calculates new end of phase time for this phase.

3.4.11. REPHAS

```
$------------------------------------------------------------------
MACRO REPHAS
$------------------------------------------------------------------
$   RETURNS TO A USER SELECTED PREVIOUSLY ENTERED PHASE
$
PRINT'WRITE 1 2 3 4 OR 5 FOR PHASE JUMP   '
INPUT INPHASE
IF(INPHASE GE .5 AND INPHASE LT 5.5) GOTO 10
PRINT'YOU CAN ONLY CHOOSE PHASES 1 TO 5'
RETURN
$
$
10  CONTINUE
$   TIME COLUMN OF PHASE IS PRESET NEG. TO INDICATE UNENTERED PHASE
IF(PHASE(INPHASE,7) GE 0) GOTO 20
PRINT'YOU CANNOT RETURN TO AN UNENTERED PHASE'
RETURN
$
$
20  CONTINUE
$   A PREVIOUSLY ENTERED PHASE HAS BEEN SELECTED
STATE=PHASE(INPHASE,ARRAY(7,1#7))
MLBS=PHASE(INPHASE,8)
MFUEL=PHASE(INPHASE,9)
CHIP=PHASE(INPHASE,10)
THRUST=PHASE(INPHASE,11)
TPHSTOP=STATE(7)+TPHASE(INPHASE)
NOWSTEP=0
NSTATE=STATE
CALL SETVAR
$   TO IDENTIFY REENTRY OF A PHASE PUT IN INPHASES TIME AND .77777
TRAJ=TRAJ&ARRAY(6)*INPHASE&STATE(7)&ARRAY(TRAJITM-7)*7.777777777
$
$   TO AVOID POSSIBLE FORWARD JUMPS, RESET TIME AS IN INT
$   BUT FOR FORWARD PHASES ONLY
IF(INPHASE GE 4.5) GOTO 40
SCR=5-INPHASE
DO 38 I=1,SCR
PHASE(INPHASE+I,7)=-1
38   CONTINUE
40   CONTINUE
$
CALL SAVTRAJ
END
```

REPHAS is used at any decision point to return to the beginning of a previously entered phase.

To recognize previously entered phases, the time column PHASE(,7) of the array PHASE is set to -1 by INI and every entering phase sets it to a non-negative time in SAVPHAS. Hence entry is permitted only into phases whose time column is non-negative and upon entry into a lower phase all higher time columns are reset to -1 by REPHAS assuming that the user will not follow an identical trajectory so that a forward jump would be meaningless.

REPHAS is the only program in this system which requires a direct user response. In FORTRAN this could be programmed as a one-argument subroutine.

5. the PRINT statement writes the string enclosed in quotes ~~when executed~~.

6. INPUT waits for user input of one number into INPHASE
   Sigma language interactive input decides from the dimension of INPHASE that one item of input is required. If more than one numbers are input, they are discarded by Sigma.

7. Check to see if INPHASE will round to the integers 1 2 3 4 or 5. Since Sigma indexing performs automatic rounding one does not have to insist on exact integers although ~~xxx~~ 99.9% of users will input exact integers.

8. error message if user input is out of the range permitted by 7

21. INPHASE determines the row of PHASE which belongs to the phase selected by the user.
    ARRAY (7, 1#7) generates a vector of 7 equally spaced components which are in the range 1 → 7 (written 1 # 7). These are the integers 1 2 3 4 5 6 7

    Hence using Sigma's multicomponent indexing we pick the first 7 components of the INPHASE row of PHASE and store them into STATE.

27, 29 we must also restore all individually named and calculated items for possible entry to RUNGE

30, 31 again make an identifying entry into TRAJ to permit easy subsequent recognition of this step

3.4.12. SAVTRAJ

```
$-----------------------------------------------------------------------
MACRO SAVTRAJ
$-----------------------------------------------------------------------
$    SAVES TRAJITM ITEMS PER POINT IN A VECTOR CALLED TRAJ
$
SCRACH=STATE
SCRACH(5)=ALT
TRAJ=TRAJ&SCRACH&G&CHIP/PI*180&MFUEL&THRUST&VELREL
END
```

SAVTRAJ decides which items to save in the trajectory vector TRAJ

The user must remember that INI SAVPHAS and REPHAS also make entries into TRAJ and each of them puts in TRAJITM components. Hence SAVTRAJ should also save TRAJITM components or conversely TRAJITM should be set (in INI) to the number of items saved by SAVTRAJ. This number should not be less than 8 because of a constraint in REPHAS.

In this case we save 12 items per point:

W
u
v
x
altitude $= (x^2+y^2+z^2)^{1/2} - R_{earth}$
time T z
G
χ_p in degrees
MFUEL
THRUST
VELREL

5. 4. 13. TRAJSHO

```
51.  $------------------------------------------------------------------
52.  MACRO TRAJSHO
 2.  $------------------------------------------------------------------
 3.  $
 4.  $   GIVES A FIRST LOOK AT THE TRAJECTORY
 5.  $
 6.  $   RESTRUCTURE TRAJ AS 2-DIM ARRAY WITH EACH STEP IN ONE ROW
 7.  SCR=NCO(TRAJ)
 8.  TRAJ=ARRAY(SCR/TRAJITM$TRAJITM,TRAJ)
 9.  PRINT TRAJ
10.  $   RESTORE TRAJ TO ITS ORIGINAL FORM (ONE-DIM. ARRAY)
11.  TRAJ=ARRAY(SCR,TRAJ)
12.  END
```

TRAJSHO in this version simply prints TRAJ as a two-dimensional array with one row per point saved.

The user may easily rewrite this to suit his convenience.

Sigma ~~fixes~~ provides automatic formatting of output and the user can only control the number of digits of the output. On the CDC computers all calculations are done in single precision floating point so that the maximum available number of significant digits is 14 to 15 decimal digits. In all subsequent outputs we print 8 digits only.

This routine uses several array handling features in Sigma and for FORTRAN it would have to be rewritten using proper formatting while in AMTRAN one would have to display reasonable pieces from the vector TRAJ.

7. finds the number of components (NCO) of the vector TRAJ

8 restructures TRAJ as a two dimensional ~~x~~ array where each row represents one saved point and each column the same item for each saved point

11 reforms the ~~x~~ vector TRAJ so that the user can proceed with CALL FLY without thinking about the restructured TRAJ of 8.

If TRAJ is left two dimensional, the concatenation of SAVTRAJ no longer works in the desired way.

For display one either displays a column of the two dimensional TRAJ or transposes TRAJ (sigma operator TP) and displays its rows.

3.4.14. INSTAG1

```
 6.  $-----------------------------------------------------------------
 7.  MACRO INSTAG1
 2.  $-----------------------------------------------------------------
 3.  $    INITIALIZES QUANTITIES CHARACTERISTIC OF ROCKET STAGE
 4.  $    FIRST STAGE
 5.  $
 6.  $
 7.  MLBS=3.5E6
 8.  $                      LBS       WEIGHT OF ROCKET
 9.  MFUEL=2277400
10.  $                      LBS       WEIGHT OF FIRST STAGE FUEL
11.  NENG=7
12.  $                                NUMBER OF ENGINES
13.  THRENG=800000
14.  $                      LBS       THRUST PER ENGINE
15.  TSEALEV=0
16.  $                                =0 IF VACUUM THRUST GIVEN,=1 FOR SEALEVEL
17.  MDOT=2081
18.  $                      LBS/SEC   WEIGHT FLOW PER ENGINE (AT FULL TROTTLE)
19.  AREANOZ=58.14
20.  $                      FT**2     NOZZLE EXIT AREA OF ONE ENGINE
21.  AREAREF=11950
22.  $                      FT**2     REFERENCE AREA FOR DRAG CALCULATIONS
23.  THRUST=THRENG*NENG
24.  G=(THRUST-FAA)/MLBS
25.  ISP=THRUST/MDOT
26.  END
```

INSTAG1 gathers together constants which belong to a stage of a vehicle so that the user can easily write his own INSTAG2 INSTG3 or INSTAG1A to initialize various other stages of the vehicle.

15. The quantity TSEALEV was included to make a logical decision in NEWTHRS and automatically provide the correct sea-level or vacuum thrust formula provided the user set TSEALEV to correspond to the type of the THRENG provided.

Since NEWTHRS is called at every step by SENHR however it was subsequently decided that the user should edit NEWTHR to correspond to his setting of THRENG, hence TSEALEV is unused at the moment.

Hence the user may make a choice here - do one more logical decision at each step or remember on the definition of each new INSTAG to make sure that NEWTHRS corresponds to the THRENG given in the new INSTAG.

```
$-------------------------------------------------------------------
MACRO INI
$-------------------------------------------------------------------
$
$   INITIATES ALL VARIABLES FOR A FRESH LAUNCH
$
$
$   GEOMETRY OF EARTH AND GRAVITY
$   ****************************
REARTH=20.925738E6
$                              FEET   RADIUS OF THE EARTH
OMEGA=7.292115E-5
$                              RAD    ANGULAR VELOCITY OF EARTH
MUGRAV=1.407656E16
$                        FEET**3/SEC**2   GRAVITATIONAL CONST.* EARTH MASS
GZERO=32.1740486
$                              FT/SEC**2  GRAVITATIONAL ACCELERATION
$
$   THE ARRAY CONTAINING THE ATMOSPHERIC DENSITY SHOULD BE FILLED
$   WITH THE PROPER VALUES FROM A SUITABLE ATMOSPHERIC MODEL
ROW=ARRAY(20)*0
$
$   GEOMETRY OF LAUNCH SITE
$   ***********************
$
AZIM=40.7*PI/180
LAT=28.5*PI/180
THETA=PI/2-LAT
$
A12=COS(AZIM)*SIN(THETA)
A22=COS(THETA)
A32=-SIN(AZIM)*SIN(THETA)
$
$   THE ROCKET ITSELF
$   ****************
$
Q=0
ALT=0
CALL INSTAG1
$
$   INITIAL STATE OF THE INTEGRATION PROCESS
$   ****************************************
$
H=.5
STATE=ARRAY(7)
STATE(1)=REARTH*OMEGA*SIN(THETA)*SIN(AZIM)
STATE(2)=0
STATE(3)=REARTH*OMEGA*SIN(THETA)*COS(AZIM)
STATE(4)=0
STATE(5)=REARTH
STATE(6)=0
STATE(7)=0
NSTATE=STATE
INPHASE=0
SIGMA=0
CALL SETVAR
```

```
$
$    TRAJECTORY AND STARTING POINTS OF EACH PHASE
$    ***********************************************
$
TRAJITM=12
TRAJ=ARRAY(TRAJITM)*0
PHASE=ARRAY(5&11)
$    SET TIME COLUMN TO -1 TO RECOGNIZE UNENTERED PHASES SEE REPHAS
PHASE(,7)=-1
$
$    CONTROL OF LOGIC FLOW OF THE FLIGHT
$    ***********************************
$
TSTOF=100
GMAX=3
QMAX=100
ALTMAX=200000
$
TPHASE=ARRAY(5)
TPHASE(1)=20
TPHASE(2)=50
TPHASE(3)=30
TPHASE(4)=40
TPHASE(5)=200
$
CHIDOT=1.3*PI/180
$                           RADIANS   RATE OF PITCHOVER IN PHASE TWO
A2=.5/180*PI
$                           RAD/SEC   CONTROL TURN RATE
SAVSTEP=2
NOWSTEP=0
$
$    TEMPORARY SETTINGS
$    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
FAA=0
FAN=0
$
CALL SAVPHAS
END
```

INI initializes all variables which are grouped by function for for easier orientation.

4.0 USE OF MILTOP

First the user should inspect and edit INI to ensure that all constants are set to values corresponding to his current problem.

If a spherical-earth-approximation is not accurate enough, he should edit INI and RHS to provide the proper G_{11} and G_{TO} instead of the simple $GRAVIT = \mu_e / r^3$ provided now.

Secondly the user should inspect and edit INSTAG1 to correspond to the stage parameters of his problem and provide INSTAG2 or INSTAG3 if it is a multistage problem.

Thirdly he should inspect and EDIT FLY to make certain that only meaningful limits are checked and included. At this point MILTOP is ready to fly and all the user has to execute is

```
CALL INI
CALL FLY
```

when processing stops TRAJ contains the trajectory flown so far and the beginnings of phases are saved in PHASE. Throttling has taken place automatically, passing of phase boundaries also automatically and if fuel is exhausted coasting is instigated automatically. One of the following conditions is true

(i) TSTOP reached by time T

(ii) one of the limits set in FLY has been reached

(iii) 200 steps have been taken without (i) or (ii)

The following options are open to the user: he can

(i) continue on the same trajectory with the same constants and conditions by CALL FLY

(ii) examine and perhaps change if necessary any or all parameters, constants or operators by resetting their values or editing the operators.

(iii) examine and display graphically and/or numerically the trajectory flown so far or any portion of it.

(iv) generate derived quantities from those saved in the trajectory and display them graphically or numerically and if necessary save them for later use.

Here care should be taken not to duplicate any of the 800 or so names already used by the system so Appendix A should be consulted before any new name is invented.

(v) delete portions of the trajectory from the trajectory vector TRAJ to keep its length in bounds or to produce a clearer graphical display

(vi) step back to the beginning of any previously entered phase by calling

CALL REPHAS

and answering 1 2 3 4 or 5 to the interactive question from REPHAS. After this call is executed user once again has full control over all options (i) → (v) and (vii) →

(vii) shut-off one engine by

NENG = NENG − 1

or any number of engines. Here some care is needed as the system does not check if the number of engines is reasonable (that is ~~nonnegative~~ a positive integer or zero).

(viii) "drop a weight" such as an empty stage or launch-escape-tower simply by reducing the mass of the vehicle

MLBS = MLBS — (dropped mass)

(ix) initiate a new stage by executing

CALL INSTAG2

Note that after all these options except REPHAS in (vi), the phase remains the same so the flight continues in this phase by resetting TSTOP or the limit which stopped the flight and executing

CALL FLY

This technique allows the user the freedom to fly the second stage in the last phase of the first stage or to provide on his own an extension to the phases by an increase of the array PHASE and some control based on INPHASE to direct the calculation of phase dependent quantities such as for example NEWCHI.

For the initial choice of control quantities the user has to choose and provide

(i) the time duration of each phase

(ii) $\dot{\chi}_p$, the rate of change of χ_p in phase two

(iii) A_2, the rate of change of χ_p in the control term of phase 5

then fly a trajectory and from the results derive a new mass of the vehicle, mass of fuel on board and perhaps new control parameters and fly again and once the first phases appear reasonable, step back and alter control variables on the last phases only.

5. MODIFICATIONS OF THE PROGRAM

The modularity of the program makes user modifications very easy. Hence it is hoped that this program will be adaptable to many uses of which payload optimization is just one.

The user can for example easily analyze the effect of various approximations in the calculation such as for example a spherical earth, or the neglect of the atmosphere altogether or at various altitudes and so on to derive a working knowledge when and when not a certain approximation is or is not warranted.

The unique set of global names is the only way in which every step of the integration can be selected to be a decision point with complete control. This set requires however that the user carefully updates Appendix A every time he defines a new name to avoid undesired name duplication which can produce side-effects without error messages.

5.1 Translation to Other Languages.

Translation to AHTRAN should be relatively simple. If AHTRAN subroutines do not automatically accept console level variables as global, all names used by MILTOP should be declared GLOBAL, hence available to all routines and the console level.

Antran graphics provides a similar capability to the Sigma Graphics on Storage tube CRT terminals.

On translation to FORTRAN, all variables should be declared in a named common (or blank common for that matter) and this common block should be declared in every routine where the Sigma macros would become subroutines without arguments in FORTRAN and the Sigma functions such as DENSITY would become FORTRAN functions.

The notation of SIGMA is sufficiently close to FORTRAN to make the translation a relatively direct affair. Any statement using a feature peculiar to SIGMA is explained in section 3.4 where each routine is described in detail.

The lack of graphic output cannot be helped in FORTRAN unless the installation has some standard plotting package available. Otherwise printer output by TRAJSHO will ~~be the~~ be the only output.

The Univac operating system seems to provide some Interactive FORTRAN capability on their 1100-series computers so the MSFC facility could use this interactive fortran to operate much like in SIGMA provided that the interactive FORTRAN permits statement by statement execution at the console and preserves the common block from one user request to the next.

If this is not the case, one can still use a FORTRAN version of MILTOP in the following way.

(i) run a simple trajectory as a batch job but at the end of the session store away all of the common block either by names or simply sequentially as an output data file

(ii) on all subsequent runs first run a small FORTRAN program which fills the 'MILTOP' common block from the previously stored data file containing the last used common block.

Even if all routines are recompiled at every try this should only be a short-fast-turn-around job. It will be even faster if all routines are compiled and stored away so that only one routine has to be run for each new try which may look like

SUBROUTINE NEXTTRY

or

or

```
MLBS = MLBS - 5000
TSTOP = 200
CALL FLY

end.
```

```
Call REPHAS (5)
A2 = A2/2
Call FLY

end
```

```
Call INBTA62
TSTOP = 300
Call FLY

end.
```

6.0 WORKED EXAMPLES

This section contains printout from an "unretouched" real-life work session where a new user attempts to familiarize himself with MILTOP.

First, the initial stages of the trajectory are calculated in batch mode processing then exploratory work proceeds on the console of the storage tube graphics terminal.

No attempt is made to devise and develop sophistic optimization techniques. Each user should first familiarize himself with the system and then develop his own style of working – some may prefer the preset batch approach, others the direct interactive graphic approach, some a mixture of both.

Because an atmospheric routine and drag coefficient tables were not available to the author all subsequent examples are run with NEWRHS redefined as the empty macro

```
MACRO NEWRHS
END
```

so that the FAA = 0 FAN = 0 preset by INI remains in force.

The work-session is explained by written comments alongside the examples.

6.1 A new trajectory "batch style"

One starts by setting the output line length to 80 characters so as to make output presentable on standard size papers.

Then one reads-in and compiles all programs. The following print-out is continuous for the next 11 pages (6-2 till 6-13).

```
!LENGTH 80
$--------------------------------------------------------------------
MACRO INI
$--------------------------------------------------------------------
$
$   INITIATES ALL VARIABLES FOR A FRESH LAUNCH
$
$
$   GEOMETRY OF EARTH AND GRAVITY
$   ******************************
REARTH=20.325738E6
$                           FEET   RADIUS OF THE EARTH
OMEGA=7.292115E-5
$                           RAD    ANGULAR VELOCITY OF EARTH
MUGRAV=1.407656E16
$                       FEET**3/SEC**2  GRAVITATIONAL CONST.* EARTH MASS
GZERO=32.1740486
$                           FT/SEC**2  GRAVITATIONAL ACCELERATION
$
$   THE ARRAY CONTAINING THE ATMOSPHERIC DENSITY SHOULD BE FILLED
$   WITH THE PROPER VALUES FROM A SUITABLE ATMOSPHERIC MODEL
ROW=ARRAY(20)*0
$
$   GEOMETRY OF LAUNCH SITE
$   ***********************
$
AZIM=40.7*PI/180
LAT=28.5*PI/180
THETA=PI/2-LAT
$
A12=COS(AZIM)*SIN(THETA)
A22=COS(THETA)
A32=-SIN(AZIM)*SIN(THETA)
$
$   THE ROCKET ITSELF
$   *****************
$
Q=0
ALT=0
CALL INSTAG1
$
```

```
$     ******************************************
$
H=.5
STATE=ARRAY(7)
STATE(1)=REARTH*OMEGA*SIN(THETA)*SIN(AZIM)
STATE(2)=0
STATE(3)=REARTH*OMEGA*SIN(THETA)*COS(AZIM)
STATE(4)=0
STATE(5)=REARTH
STATE(6)=0
STATE(7)=0
NSTATE=STATE
INPHASE=0
SIGMA=0
CALL SETVAR
$
$   TRAJECTORY AND STARTING POINTS OF EACH PHASE
$     ******************************************
$
TRAJITM=12
TRAJ=ARRAY(TRAJITM)*0
PHASE=ARRAY(5&11)
$   SET TIME COLUMN TO -1 TO RECOGNIZE UNENTERED PHASES SEE REPHAS
PHASE(,7)=-1
$
$   CONTROL OF LOGIC FLOW OF THE FLIGHT
$   ******************************
$
TSTOP=100
GMAX=3
QMAX=100
ALTMAX=200000
$
TPHASE=ARRAY(5)
TPHASE(1)=20
TPHASE(2)=50
TPHASE(3)=30
TPHASE(4)=40
TPHASE(5)=200
$
CHIDOT=1.3*PI/180
$                              RADIANS   RATE OF PITCHOVER IN PHASE TWO
A2=.5/180*PI
$                              RAD/SEC   CONTROL TURN RATE
SAVSTEP=2
NOWSTEP=0
$
$   TEMPORARY SETTINGS
$   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
FAA=0
FAN=0
$
CALL SAVPHAS
END
$
$
$------------------------------------------------------------------
MACRO INSTAG1
$------------------------------------------------------------------
$   INITIALIZES QUANTITIES CHARACTERISTIC OF ROCKET STAGE
$   FIRST STAGE
$
$
MLBS=3.5E6
$                              LBS          WEIGHT OF ROCKET
```

```
$                                LBS        WEIGHT OF FIRST STAGE FUEL
NENG=7
$                                           NUMBER OF ENGINES
THRENG=800000
$                                LBS        THRUST PER ENGINE
TSEALEV=0
$                                           =0 IF VACUUM THRUST GIVEN,=1 FOR SEALEVE
MDOT=2081
$                                LBS/SEC    WEIGHT FLOW PER ENGINE (AT FULL TROTTLE)
AREANOZ=58.14
$                                FT**2      NOZZLE EXIT AREA OF ONE ENGINE
AREAREF=11950
$                                FT**2      REFERENCE AREA FOR DRAG CALCULATIONS
THRUST=THRENG*NENG
G=(THRUST-FAA)/MLBS
ISP=THRUST/MDOT
END
$
$
$-----------------------------------------------------------------------
MACRO SETVAR
$-----------------------------------------------------------------------
$    SET OR CALCULATE ALL INDIVIDULLY NAMED VARIABLES
$    EXPECTS THE CURRENT STATE VECTOR IN  NSTATE
$
W=NSTATE(1)
U=NSTATE(2)
V=NSTATE(3)
X=NSTATE(4)
Y=NSTATE(5)
Z=NSTATE(6)
T=NSTATE(7)
$
ALT=SQRT(X*X+Y*Y+Z*Z)-REARTH
$
WREL=W-OMEGA*(A22*Z-A32*Y)
UREL=U-OMEGA*(A32*X-A12*Z)
VREL=V-OMEGA*(A12*Y-A22*X)
$
VELREL=SQRT(WREL*WREL+UREL*UREL+VREL*VREL)
$
$
CALL NEWCHI
CALL NEWDRAG
CALL NEWTHRS
$
END
$
$
$-----------------------------------------------------------------------
MACRO NEWDRAG
$-----------------------------------------------------------------------
$
$    CALCULATES THE DRAG FORCES
$                         FAA  IS THE AXIAL DRAG
$                         FAN  IS THE NORMAL DRAG
$
Q=0.5*DENSITY(ALT)*(W*W+U*U+V*V)
FAA=Q*AREAREF*CA(MACH(ALT))
FAN=0
END
$
$-----------------------------------------------------------------------
FUNCTION MACH(ALTITUD)
$------
```

```
$    SUCH A TABLE IS NOT AVAILABLE TO THE AUTHOR
MACH=0
END
$
$-----------------------------------------------------------------
FUNCTION CA(MACHNUM)
$-----------------------------------------------------------------
$    AGAIN A TABLE LOOKUP ON A TABLE UNAVAILABLE TO THE AUTHCR
CA=0
$
END
$
$-----------------------------------------------------------------
FUNCTION DENSITY(ALTITUD)
$-----------------------------------------------------------------
$    LOOKS UP THE DENSITY IN A TABLE IN STEPS OF 10,000 FEET
$    ABOVE 200,000 FEET RETURNS ZERO AS OUTSIDE THE ATMOSPHERE
$
IF(ALTITUD LE 200000) GOTO 10
DENSITY=0
RETURN
$
10   CONTINUE
GLOBAL ROW
DENSITY=ROW(ALTITUD/10000)
END
$
$
$-----------------------------------------------------------------
MACRO NEWCHI
$-----------------------------------------------------------------
IF(INPHASE NE 0) GOTO 1
$    FIRST INITIALIZATION
CHIP=0
CHIY=0
RETURN
$
1  CONTINUE
IF(INPHASE NE 1) GOTO 2
$    PHASE  ONE
$    VERTICAL RISE
CHIP=0
CHIY=0
RETURN
$
2  CONTINUE
IF(INPHASE NE 2) GOTO 3
$    PHASE  TWO
$    TILT-OVER, STARTING TIME CF PHASE IN PHASE(INPHASE,7)
CHIP=CHIDOT*(T-PHASE(INPHASE,7))
CHIY=0
RETURN
$
3  CONTINUE
IF(INPHASE NE 3) GOTO 4
$    PHASE  THREE
$    GRAVITY TURN
CHIP=ATAN2(WREL,UREL)
CHIY=0
RETURN
$
4  CONTINUE
IF(INPHASE NE 4) GOTO 5
$    PHASE  FOUR
$    CUT FREEZE LEAVES CHIP AND CHIY
```

(6-6

```
RETURN
$
5  CONTINUE
IF(INPHASE NE 5) PRINT'INPHASE INCORRECT'
$   PHASE  FIVE
$   CONTROL TURN
CHIP=A1+A2*(T-PHASE(INPHASE,7))
CHIY=0
$
END
$
$
$------------------------------------------------------------
MACRO REPHAS
$------------------------------------------------------------
$   RETURNS TO A USER SELECTED PREVIOUSLY ENTERED PHASE
$
PRINT'WRITE 1 2 3 4 OR 5 FOR PHASE JUMP    '
INPUT INPHASE
IF(INPHASE GE .5 AND INPHASE LT 5.5) GOTO 10
PRINT'YOU CAN ONLY CHOOSE PHASES 1 TO 5'
RETURN
$
$
10  CONTINUE
$   TIME COLUMN OF PHASE IS PRESET NEG. TO INDICATE UNENTERED PHASE
IF(PHASE(INPHASE,7) GE 0) GOTO 20
PRINT'YOU CANNOT RETURN TO AN UNENTERED PHASE'
RETURN
$
$
20  CONTINUE
$   A PREVIOUSLY ENTERED PHASE HAS BEEN SELECTED
STATE=PHASE(INPHASE,ARRAY(7,1#7))
MLBS=PHASE(INPHASE,8)
MFUEL=PHASE(INPHASE,9)
CHIP=PHASE(INPHASE,10)
THRUST=PHASE(INPHASE,11)
TPHSTOP=STATE(7)+TPHASE(INPHASE)
NOWSTEP=0
NSTATE=STATE
CALL SETVAR
$   TO IDENTIFY REENTRY OF A PHASE PUT IN INPHASES TIME AND .77777
TRAJ=TRAJ&ARRAY(6)*INPHASE&STATE(7)&ARRAY(TRAJITM-7)*7.777777777
$
$   TO AVOID POSSIBLE FORWARD JUMPS, RESET TIME AS IN INT
$   BUT FOR FORWARD PHASES ONLY
IF(INPHASE EQ 5) GOTO 40
SCR=5-INPHASE
DO 38 I=1,SCR
PHASE(INPHASE+I,7)=-1
38   CONTINUE
40   CONTINUE
$
CALL SAVTRAJ
END
$
$
$------------------------------------------------------------
MACRO SAVPHAS
$------------------------------------------------------------
$   SAVES THE STATE OF THE SYSTEM AT TE BEGINNING OF EACH PHASE
$   SO CALL REPHAS CAN RESTART TRAJECTORY FROM THIS POINT ONWARDS.
$
INPHASE=INPHASE+1
```

```
$   START OF A NEW PHASE AND THE TIME OF START OF IT
SCR=INPHASE*ARRAY(TRAJITM)
SCR(7)=STATE(7)
TRAJ=TRAJ&SCR
$   SET CONSTANT A1 FOR CONTROL TURN IN PHASE FIVE
IF(INPHASE EQ 5) A1=CHIP
CALL SAVTRAJ
PHASE(INPHASE,ARRAY(7,1#7))=STATE
PHASE(INPHASE,8)=MLBS
PHASE(INPHASE,9)=MFUEL
PHASE(INPHASE,10)=CHIP
PHASE(INPHASE,11)=THRUST
TPHSTOP=STATE(7)+TPHASE(INPHASE)
END
$
$
$-------------------------------------------------------------
MACRO NEWTHRS
$-------------------------------------------------------------
IF(G LT GMAX) RETURN
$   REDUCE MDOT AND RECALCULATE THROTTLED THRUST
THRUST=GMAX*MLBS+FAA
MDOT=THRUST/ISP
END
$
$
$-------------------------------------------------------------
MACRO RUNGE
$-------------------------------------------------------------
$   ON ENTRY STATE AND IDIVIDUAL VARIABLES SHOULD MATCH
$   DURINT RUNGE STATE IS UNTOUCHED TILL THE END
$   AT END STATE IS UPDATED TOGETHER WITH THE INDIVIDUAL VARIABLES
$   NOTE THAT MLBS AND WHEN THROTTLED MDOT THRUST ARE FNS OF TIME
$
CALL NEWRHS
K1=H*RHS
$
NSTATE=STATE+K1/2
CALL SETVAR
MLBS=MLBS-NENG*MDOT*H/2
CALL NEWRHS
K2=H*RHS
$
NSTATE=STATE+K2/2
CALL SETVAR
CALL NEWRHS
K3=H*RHS
$
NSTATE=STATE+K3
CALL SETVAR
MLBS=MLBS-NENG*MDOT*H/2
CALL NEWRHS
K4=H*RHS
$
$
STATE=K1/6+K2/3+K3/3+K4/6+STATE
NSTATE=STATE
CALL SETVAR
MFUEL=MFUEL-NENG*MDOT*H
G=(THRUST-FAA)/MLBS
END
$
$
$
$
```

(6-8

```
MACRO DERIVS
$-----------------------------------------------------------------------
COSCHIP=COS(CHIP)
SINCHIP=SIN(CHIP)
SIGMA=ATAN2(VREL,WREL*COSCHIP-UREL*SINCHIP)
GRAVIT=MUGRAV/((X*X+Y*Y+Z*Z)**1.5)
DW=(-COSCHIP*FAN*COS(SIGMA)+SINCHIP*(THRUST-FAA))/MLBS*GZERO-X*GRAVI
DU=(SINCHIP*FAN*COS(SIGMA)+COSCHIP*(THRUST-FAA))/MLBS*GZERO-Y*GRAVI
DV=-FAN*SIN(SIGMA)/MLBS*GZERO-Z*GRAVIT
$
RHS=DW&DU&DV&W&U&V&1
END
$
$
$-----------------------------------------------------------------------
MACRO FLY
$-----------------------------------------------------------------------
$   ADVANCES TRAJECTORY EITHER BY 200 STEPS OR TILL ONE OF
$   THE USER SPECIFIED LIMITS IS EXCEEDED
$
DO 10 I=1,200
$   ENTER A NEW PHASE AUTOMATICALLY WHEN NECESSARY
IF(T LT TPHSTOP) GOTO 5
$   AT OR OVER PHASE STOP,SAVPHAS, THEN SET ALL VAR. THAT CHANGE
CALL SAVPHAS
CALL NEWCHI
5  CONTINUE
IF(T GE TSTOP) GOTO 100
IF(Q GT QMAX) GOTO 100
IF(ALT GT ALTMAX) GOTO 100
IF(MFUEL LE 0) CALL NOFUEL
$
CALL RUNGE
$
NOWSTEP=NOWSTEP+1
IF(MOD(NOWSTEP,SAVSTEP) EQ 0) CALL SAVTRAJ
10 CONTINUE
$
$
PRINT'YOU HAVE COMPLETED 200 STEPS WITHOUT REACHING ANY LIMIT'
$
100 CONTINUE
PRINT T,TSTOP,G,GMAX,Q,QMAX,ALT,ALTMAX,MFUEL
CALL TRAJSHO
END
$
$-----------------------------------------------------------------------
MACRO NOFUEL
$-----------------------------------------------------------------------
$
$   CORRECT MLBS FOR EXCESS FUEL SUBTRACTED IN LAST FUELED STEP
IF(MFUEL LT 0) MLBS=MLBS+MFUEL
MFUEL=0
$
$   ALLOW THE ROCKET TO COAST
$
MDOT=0
THRUST=0
G=0
END
$-----------------------------------------------------------------------
MACRO SAVTRAJ
$-----------------------------------------------------------------------
$
$   SAVES TRAJITM ITEMS PER POINT IN A VECTOR CALLED TRAJ
```

```
SCRACH=STATE
SCRACH(5)=ALT
TRAJ=TRAJ&SCRACH&G&CHIP/PI*180&MFUEL&THRUST&VELREL
END
$
$
$--------------------------------------------------------------------------------
MACRO TRAJSHO
$--------------------------------------------------------------------------------
$
$   GIVES A FIRST LOOK AT THE TRAJECTORY
$
$   RESTRUCTURE TRAJ AS 2-DIM ARRAY WITH EACH STEP IN ONE ROW
SCR=NCO(TRAJ)
TRAJ=ARRAY(SCR/TRAJITM&TRAJITM,TRAJ)
PRINT TRAJ
$   RESTORE TRAJ TO ITS ORIGINAL FORM (ONE-DIM. ARRAY)
TRAJ=ARRAY(SCR,TRAJ)
END
$
$
$
$
$
```

USER INPUT

```
MACRO NEWDRAG
$   IGNORES THE ATMOSPHERE
$   LEAVES DRAG PRESET TO ZERO
END
```

here the user redefines (and overwrites previous def.) NEWDRAG to ignore the atmosphere

```
$
$
CALL INI
TSTOP=250
H=2
CALL FLY
```

calls initialization

sets new TSTOP at 250 seconds

sets new step size at 2 seconds

```
4000. STATEMENTS EXECUTED.TYPE GO OR QUIT..
4000. STATEMENTS EXECUTED.TYPE GO OR QUIT..
4000. STATEMENTS EXECUTED.TYPE GO OR QUIT..
```

Sigma reminds the user how many statements he is executing (in this case ≥ 12,000

output from FLY

```
T=  156.00000
TSTOP=  250.00000
G=  3.0000000
GMAX=  3.0000000
Q= 0.
QMAX=  100.00000
ALT=  201957.28
ALTMAX=  200000.00
MFUEL=  105178.77
NCO(TRAJ)=    50    12
TRAJ=
```

trajectory stopped at t=156 which is <250 hence a limit was reached

altitude exceeded 200,000 feet.

we see from SAVTRAJ above that each line contains

W U V X ALT Z T G X_p MFUEL THRUST VELREL

output from TRAJSHO

```
0.              0.              0.              0.              0.
0.              0.              0.              0.              0.
0.              0.

1.0000000       1.0000000       1.0000000       1.0000000       1.0000000
1.0000000       0.              1.0000000       1.0000000       1.0000000
1.0000000       1.0000000

874.47205       0.              1016.5676       0.              0.
0.              0.              1.6000000       0.              2277400.0
5600000.0       .36379788E-11

874.46131       79.061470       1016.6551       3497.8739       157.64779
4056.6536       4.0000000       1.6270878       0.              2219132.0
5600000.0       79.405444

874.42907       161.67249       1016.6176       6995.6618       639.97404
```

line put in by SAVPHAS to indicate entry of phase 1.

(C-10

5600000.0	162.36044			
874.37534	247.96032	1016.5551	10493.278	1461.4289
12199.561	12.000000	1.6841113	0.	2102596.0
5600000.0	248.99229			
874.30014	338.05897	1016.4677	13990.636	2636.9848
16265.615	16.000000	1.7141486	0.	2044328.0
5600000.0	339.43498			
874.20347	432.10961	1016.3553	17487.650	4182.1637
20331.269	20.000000	1.7452768	0.	1986060.0
5600000.0	433.82973			
2.0000000	2.0000000	2.0000000	2.0000000	2.0000000
2.0000000	$\boxed{20.000000}$	2.0000000	2.0000000	2.0000000
2.0000000	2.0000000			
874.20347	432.10961	1016.3553	17487.650	4182.1637
20331.269	20.000000	1.7452768	0.	1986060.0
5600000.0	433.82973			
884.39584	529.94866	1016.2180	20997.944	6112.7697
24396.424	24.000000	1.7775565	5.2000000	1927792.0
5600000.0	532.08455			
915.61516	630.13889	1016.0557	24590.859	8441.5012
28450.980	28.000000	1.8110529	10.400000	1869524.0
5600000.0	633.80028			
968.39855	730.85542	1015.8686	28351.624	11174.296
32524.837	32.000000	1.8458358	15.600000	1811256.0
5600000.0	739.40775			
1043.1377	830.19925	1015.6567	32367.329	14309.707
36587.896	36.000000	1.8819810	20.800000	1752988.0
5600000.0	849.85856			
1140.0714	926.20749	1015.4199	36726.326	17838.636
40650.057	40.000000	1.9195700	26.000000	1694720.0
5600000.0	966.32123			
1259.2780	1016.8642	1015.1584	41517.605	21744.114
44711.222	44.000000	1.9586912	31.200000	1636452.0
5600000.0	1090.0291			
1400.6696	1100.1115	1014.8721	46830.137	26001.122
48771.291	48.000000	1.9994402	36.400000	1578184.0
5600000.0	1222.1916			
1563.9863	1173.8621	1014.5612	52752.203	30576.471
52830.166	52.000000	2.0419206	41.600000	1519916.0
5600000.0	1363.9391			
1748.7916	1236.0108	1014.2257	59370.687	35428.726
56887.748	56.000000	2.0862454	46.800000	1461648.0
5600000.0	1516.2877			
1954.4685	1284.4478	1013.8657	66770.373	40508.184
60943.939	60.000000	2.1325372	52.000000	1403380.0
5600000.0	1680.1211			
2180.2162	1317.0712	1013.4811	75033.205	45756.904
64398.641	64.000000	2.1809299	57.200000	1345112.0
5600000.0	1856.1824			

phase 2 starts here t f = 20 sec

	2425.0485	1331.8008	1013.0721	[illegible]	[illegible]
	69051.755	68.000000	2.2315700	62.400000	1286844.0
	5600000.0	2045.0753			
ase 3	3.0000000	3.0000000	3.0000000	3.0000000	3.0000000
here	3.0000000	$\boxed{70.000000}$	3.0000000	3.0000000	3.0000000
	3.0000000	3.0000000			
t=70 sec	2554.2620	1331.8145	1012.8584	89216.134	53800.569
	71077.687	70.000000	2.2577823	65.000000	1257710.0
	5600000.0	2144.4858			
	2669.1700	1357.7982	1012.6387	94439.001	55520.214
	73103.185	72.000000	2.2846177	52.662963	1228576.0
	5600000.0	2251.1006			
	2909.0445	1405.1635	1012.1809	105590.99	62115.732
	77152.832	76.000000	2.3402483	55.144487	1170308.0
	5600000.0	2472.9732			
	3162.0945	1446.4252	1011.6989	117728.92	67900.647
	81200.600	80.000000	2.3986568	57.473066	1112040.0
	5600000.0	2706.5326			
	3428.1027	1481.6616	1011.1926	130905.02	73852.713
	85246.391	84.000000	2.4600549	59.656849	1053772.0
	5600000.0	2951.9817			
	3706.9304	1510.9858	1010.6623	145170.83	79950.339
	89290.109	88.000000	2.5246787	61.704321	995504.00
	5600000.0	3209.5406			
	3998.5194	1534.5354	1010.1079	160577.48	86172.737
	93331.657	92.000000	2.5927895	63.624007	937236.00
	5600000.0	3479.4547			
	4302.8928	1552.4648	1009.5293	177176.03	92500.033
	97370.939	96.000000	2.6646770	65.424257	878968.00
	5600000.0	3762.0027			
	4620.1560	1564.9386	1008.9269	195017.80	98913.351
	101407.86	100.00000	2.7406646	67.113105	820700.00
	5600000.0	4057.5031			
Phase 4	4.0000000	4.0000000	4.0000000	4.0000000	4.0000000
starts here	4.0000000	$\boxed{100.00000}$	4.0000000	4.0000000	4.0000000
at t=100 sec	4.0000000	4.0000000			
	4620.1560	1564.9386	1008.9269	195017.80	98913.351
	101407.86	100.00000	2.7406646	67.113105	820700.00
	5600000.0	4057.5031			
	4948.5865	1576.7747	1008.3006	214152.15	105401.04
	105442.32	104.00000	2.8211132	67.113105	762432.00
	5600000.0	4366.2316			
	5286.7209	1592.8378	1007.6504	234619.43	111977.30
	109474.23	108.00000	2.9064275	67.113105	704164.00
	5600000.0	4688.2052			
	5635.1547	1613.3839	1006.9765	256459.64	118663.90
	113503.49	112.00000	2.9970629	67.113105	645896.00
	5600000.0	5023.5238			
	5991.5951	1637.4502	1006.2790	279713.01	125433.11

```
 5432298.5     5369.2662

 6348.0993     1661.6890     1005.5578     304392.45     132445.80
 121553.69     120.00000     3.0000000     67.113105     532851.01
 5264378.2     5716.5084

 6703.9823     1685.8166     1004.8132     330495.26     139556.32
 125574.44     124.00000     3.0475649     67.113105     478926.73
 5182535.1     6064.2690

 7059.4727     1709.9350     1004.0450     358022.70     146821.08
 129592.17     128.00000     3.0000000     67.113105     427075.54
 4944255.1     6412.6068

 7415.4985     1734.4417     1003.2536     386972.70     154245.31
 133606.77     132.00000     3.0000000     67.113105     376425.69
 4791421.0     6762.3699

 7770.1755     1758.5467     1002.4389     417343.64     161834.75
 137618.17     136.00000     3.0000000     67.113105     327727.53
 4643598.4     7111.4107

 8124.6686     1782.7478     1001.6010     449132.92     169594.97
 141626.25     140.00000     3.0000000     67.113105     280531.79
 4500336.4     7460.8851
```

are 5 sets here t=140 sec.

```
 5.0000000     5.0000000     5.0000000     5.0000000     5.0000000
 5.0000000     140.00000     5.0000000     5.0000000     5.0000000
 5.0000000     5.0000000

 8124.6686     1782.7478     1001.6010     449132.92     169594.97
 141626.25     140.00000     3.0000000     67.113105     280531.79
 4500336.4     7460.8851

 8482.2410     1801.0569     1000.7400     482343.71     177524.61
 145630.94     144.00000     3.0475649     69.113105     234433.78
 4430371.5     7812.4799

 8844.2018     1806.7326     999.85600     516995.60     185590.68
 149632.14     148.00000     3.0000000     71.113105     190107.98
 4226674.1     8156.0410

 9211.0674     1799.9239     998.94913     553104.83     193749.44
 153629.76     152.00000     3.0000000     73.113105     146809.15
 4095021.4     8522.3881

 9580.4358     1779.8783     998.01944     590686.18     201957.28
 157623.71     156.00000     3.0000000     75.113105     105178.77
 3969653.0     8879.3119
```

here trajectory stops at t=156 sec, ALT= 201,957.28 feet when VELREL is 8879.3119 feet/sec, because ALTMAX was set to 200 000 feet.

```
 76.  GO
******** GO *UNDEFINED NAME OR SYSTEM FUNCTION NAME
  1.  GO
******** GO *UNDEFINED NAME OR SYSTEM FUNCTION NAME
  1.  GO
******** GO *UNDEFINED NAME OR SYSTEM FUNCTION NAME
  1.  GO
```

In batch mode Sigma requires a card with Go for each 4000 statements executed so these are unused Go's

******** GO *UNDEFINED NAME OR SYSTEM FUNCTION NAME (6-13

1. !LOGOUT here the state of the system is saved for next time.

Sigma like AMTRAN provides a fixed automatic format which saves the user considerable trouble in designing format statements but is obviously not as flexible as FORTRAN for outputting tables.

In Sigma the user can control only the line length (set to 80 here) and the number of significant digits shown (set to 8 here). Hence a 12 number row is printed in three lines and on each line we have

W (x-velocity) U (y-velocity) V (z-velocity) X ALTitude ~~Z~~

Z Time G-force ratio X_p in degrees MFUEL

THRUST lbs VELREL ft/sec.

Note that INI has inserted a marker line of all zeroes and SAVTRAJ inserts a line of INPHASE's every time a new phase is entered so that each phase entry point is easily found.

Also note that the seventh (time) position of each phase marker line gives the time of phase entry. This has two advantages

(i) the exact time of phase entry is known although the printing of each SAVPHAS'th step may have jumped over it

(ii) in subsequent graphic output plotting against the time column will draw vertical lines at phase boundaries because all variables will dip to 1, 2 – 5 while time remains continuous if we plot ~~the~~ whole ~~the~~ columns regardless of phase boundaries.

6.2 Continuation of trajectory - batch style

Our user has decided to continue the trajectory to $t = 250$. He therefore resets ALTMAX = 300000 to a higher limit and leaves TSTOP = 250.

He also decides to reduce the altitude by turning the vehicle faster in the control turn of phase 5. He therefore steps back to the start of phase 5 (using REPHAS) and sets A2 = 2 * A2 for $\chi_p = a_1 + a_2 (t - t_{start,plc})$

He now loggs into the Sigma workspace saved in section 6.1 and gives it the following instructions.

```
    1.  CALL REPHAS
 WRITE 1 2 3 4 OR 5 FOR PHASE JUMP     <- here a 5 is read from the next input card
    2.  A2=2*A2                           ~~but not shown in the output~~
    3.  PRINT A2
 A2=  .17453293E-01
    4.  ALTMAX=300000
    5.  TSTOP=250
    6.  CALL FLY
    4000. STATEMENTS EXECUTED.TYPE GO OR QUIT..  } system does ≥ 8000
    4000. STATEMENTS EXECUTED.TYPE GO OR QUIT..  } statements (and uses 2 go card)
 T=  250.00000
 TSTOP=  250.00000   — T has now reached t = 250 sec
 G= 0.
 GMAX=  3.0000000
 Q= 0.
 QMAX=  100.00000
 ALT=  274440.48     <- ALTitude is 274440 feet
 ALTMAX=  300000.00
 MFUEL= 0.           <- vehicle is coasting and out of fuel
 NCO(TRAJ)=    79    12
 TRAJ=
  0.              0.              0.              0.              0.
  0.              0.              0.              0.              0.
  0.              0.

  1.0000000       1.0000000       1.0000000       1.0000000       1.0000000
  1.0000000       0.              1.0000000       1.0000000       1.0000000
  1.0000000       1.0000000

  874.47205       0.              1016.6676       0.              0.
  0.              0.              1.6000000       0.              2277400.0
  5600000.0       .36379788E-11

  874.46131       79.061470       1016.6551       3497.8739       157.64779
  4066.6536       4.0000000       1.6270878       0.              2219132.0
  5600000.0       79.405444

  874.42907       161.67249       1016.6176       6995.6618       639.97404
  8133.2072       8.0000000       1.6551085       0.              2160864.0
  5600000.0       162.36044
```

here the same trajectory as in 2.1 is printed so we do not reproduce it here but continue only with the newly calculated part

we restart the printout at the first entry to phase 5 which is identical to the printout of section 2.1. This run terminated at $t = 156$

Next REPHAS inserted its marker line, the 7's at the end indicating that it is a step back to the beginning of phase 5 and we can check that all resettable items are correctly reset.

Note that G is held at 3 and THRUST is dropping to hold it there ever since GMAX was reached at $112 \leq t \leq 116$ sec.

Once more remember that the numerical output is in each line.

	W / Z / 6-12 THRUST	U / Time / VELREL	V / G-force ratio	X / x_p	ALTitude / MFUEL
As on page 6-12 phase 5 starts at $t = 140$	5.0000000	5.0000000	5.0000000	5.0000000	5.0000000
	5.0000000	140.00000	5.0000000	5.0000000	5.0000000
	5.0000000	5.0000000			
	8124.6686	1782.7478	1001.6010	449132.92	169594.97
	141626.25	140.00000	3.0000000	67.113105	280531.79
	4500336.4	7460.8851			
	8482.2410	1801.0569	1000.7400	482343.71	177524.61
	145630.94	144.00000	3.0475649	69.113105	234433.78
	4430371.5	7812.4799			
	8844.2018	1806.7326	999.85600	516995.60	185590.68
	149632.14	148.00000	3.0000000	71.113105	190107.98
	4226674.1	8166.0410			
old trajectory of p.6-9 to 6-12 ends here	9211.0674	1799.9239	998.94913	553104.83	193749.44
	153629.76	152.00000	3.0000000	73.113105	146809.15
	4096021.4	8522.3881			
	9580.4358	1779.8783	998.01944	590686.18	201957.28
	157623.71	156.00000	3.0000000	75.113105	105178.77
	3969653.0	8879.3119			
restarted phase 5 again starts at $t = 140$	5.0000000	5.0000000	5.0000000	5.0000000	5.0000000
	5.0000000	140.00000	7.7777778	7.7777778	7.7777778
	7.7777778	7.7777778			
	8124.6686	1732.7478	1001.6010	449132.92	169594.97
	141626.25	140.00000	3.0000000	67.113105	280531.79
	4500336.4	7460.8851			
new trajectory	8434.6619	1794.7121	1000.7400	482347.00	177516.27
	145630.94	144.00000	3.0475649	71.113105	234433.78
	4430371.5	7813.3520			
	8852.9739	1781.0935	999.85600	517020.20	185523.30
	149632.14	148.00000	3.0000000	75.113105	190107.98
	4226674.1	8168.8727			
	9228.7826	1741.5621	998.94911	553181.77	193520.04
	153629.76	152.00000	3.0000000	79.113105	[illegible]

	9608.2435	1675.2768	998.01937	590854.00	201409.41
	157623.71	156.00000	3.0000000	83.113105	105178.77
	3969653.0	8885.8668			
	9990.7215	1582.2371	997.06684	630050.74	209092.93
	161613.89	160.00000	3.0000000	87.113105	64832.753
	3847183.2	9244.7201			
	10374.327	1462.2798	996.09155	670780.27	216471.94
	165600.21	164.00000	3.0000000	91.113105	25731.468
	3728491.8	9603.1343			
	10757.163	1315.3721	995.09353	713043.32	223447.55
	169582.59	168.00000	3.0000000	95.113105	-12163.483
	3613462.3	9960.3648			
fuel depleted	10752.793	1189.6078	994.07278	756063.31	229980.50
	173560.93	172.00000	0.	99.113105	0.
	0.	9939.6208			
	10748.171	1063.9280	993.02931	799065.33	236097.60
	177535.14	176.00000	0.	103.11310	0.
	0.	9920.1602			
	10743.298	938.32820	991.96313	842048.35	241799.06
	181505.13	180.00000	0.	107.11310	0.
	0.	9901.9877			
	10738.173	812.80384	990.87423	885011.37	247085.08
	185470.81	184.00000	0.	111.11310	0.
	0.	9885.1075			
	10732.798	687.35050	989.76260	927953.40	251955.85
	189432.10	188.00000	0.	115.11310	0.
	0.	9869.5237			
	10727.171	561.96370	988.62825	970873.42	256411.53
	193388.88	192.00000	0.	119.11310	0.
	0.	9855.2401			
	10721.294	436.63901	987.47116	1013770.4	260452.30
	197341.09	196.00000	0.	123.11310	0.
	0.	9842.2603			
	10715.166	311.37199	986.29132	1056643.4	264078.28
	201288.62	200.00000	0.	127.11310	0.
	0.	9830.5873			
	10708.788	186.15822	985.08870	1099491.4	267289.60
	205231.39	204.00000	0.	131.11310	0.
	0.	9820.2242			
	10702.158	60.993282	983.86330	1142313.4	270086.38
	209169.30	208.00000	0.	135.11310	0.
	0.	9811.1736			
	10695.279	-64.127234	982.61507	1185108.4	272468.72
	213102.27	212.00000	0.	139.11310	0.
	0.	9803.4376			
	10688.148	-189.20772	981.34400	1227875.3	274436.69
	217030.19	216.00000	0.	143.11310	0.
	0.	9797.0183			

```
 10680.767    -314.25258    980.05006    1270613.2    [illegible]
 220952.99     220.00000    0.           147.11310    0.
0.             9791.9173

 10673.134    -439.26618    978.73320    1313321.1    277129.81
 224870.56     224.00000    0.           151.11310    0.
0.             9788.1360

 10665.251    -564.25292    977.39339    1355998.0    277855.05
 228782.82     228.00000    0.           155.11310    0.
0.             9785.6753

 10657.116    -689.21717    976.03058    1398642.8    278166.12
 232689.68     232.00000    0.           159.11310    0.
0.             9784.5359

 10648.729    -814.16329    974.64474    1441254.5    278063.03
 236591.04     236.00000    0.           163.11310    0.
0.             9784.7180

 10640.091    -939.09566    973.23581    1483832.3    277545.76
 240486.81     240.00000    0.           167.11310    0.
0.             9786.2218

 10631.200    -1064.0186    971.80374    1526374.9    276614.32
 244376.89     244.00000    0.           171.11310    0.
0.             9789.0469

 10622.057    -1188.9366    970.34848    1568881.5    275268.65
 248261.21     248.00000    0.           175.11310    0.
0.             9793.1925


    7.  !LOGOUT
```

Again the batch mode run logs itself out to save the workspace for subsequent graphical continuation of this flight.

6.3 Interactive Graphical Displays.

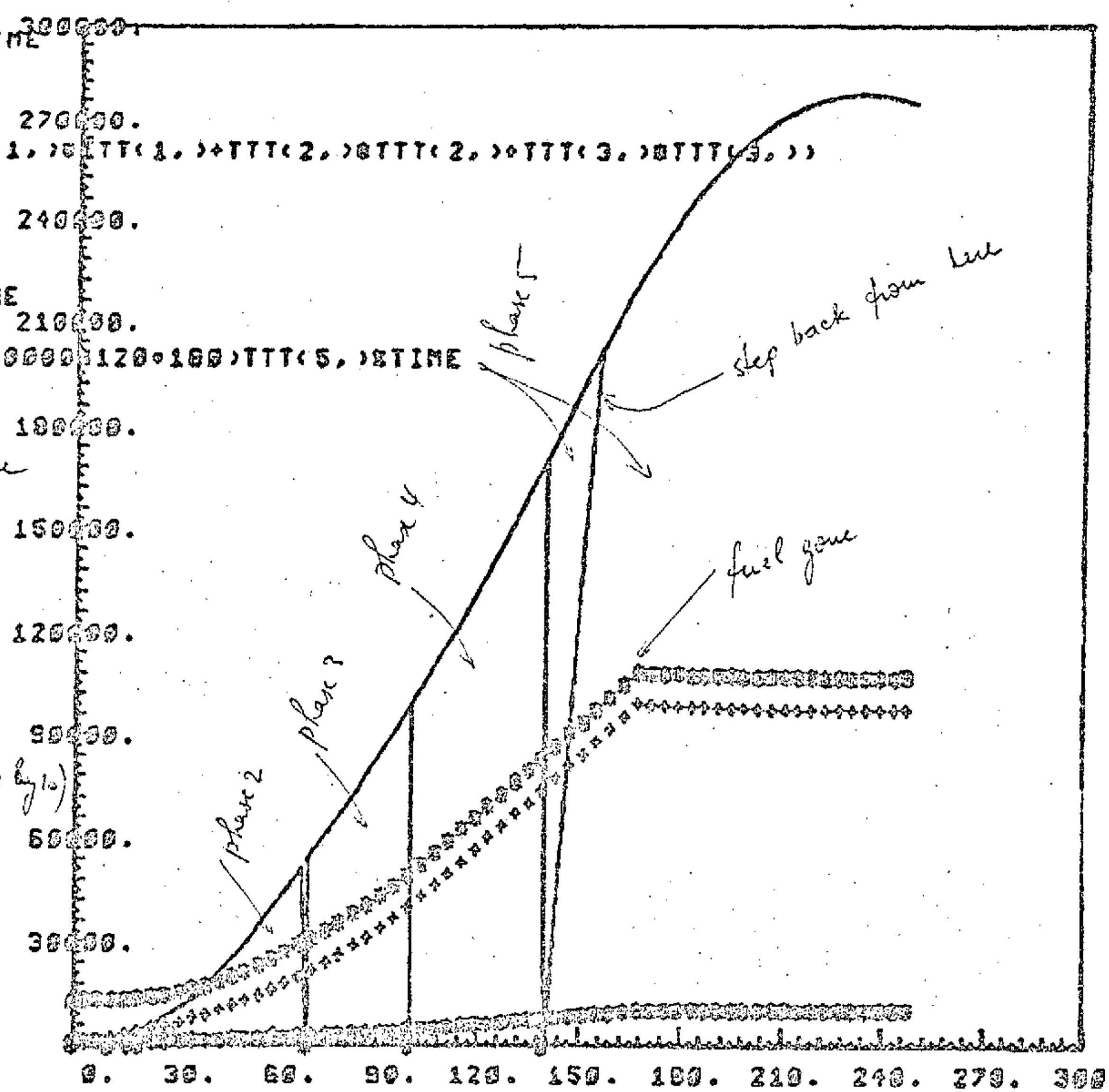
After logging into the workspace saved in section 6.2 the user decides to display the ALTitude as a function of the time.

First he rearranges TRAJ into a two dimensional array TTT whose rows represent the saved items so that for example time is TTT(7;) and for convenience he also defines TIME = TTT(7;).

here the user has displayed the altitude a function of time as a solid curve, setting the altitude window at 0 → 300 000 ft and the time window 0 to 300 sec.

```
10.  [+]TTT(12.)*10*TIME

VELINRT=SQRT(TTT(1.)*TTT(1.)+TTT(2.)*TTT(2.)+TTT(3.)*TTT(3.))

10.  [*]VELRELSTIME
         INRTSTIME

[*]VELINRT*10*TIME

DISPLAY(150000*200000*120*180)TTT(5.)*TIME
```

backspace



using + we display on the same picture TTT(12,) * 10 which is VELREL * 10

next we calculate the inertial velocity and display it using * to get a curve close to the axis (we forgot to multiply by 10)

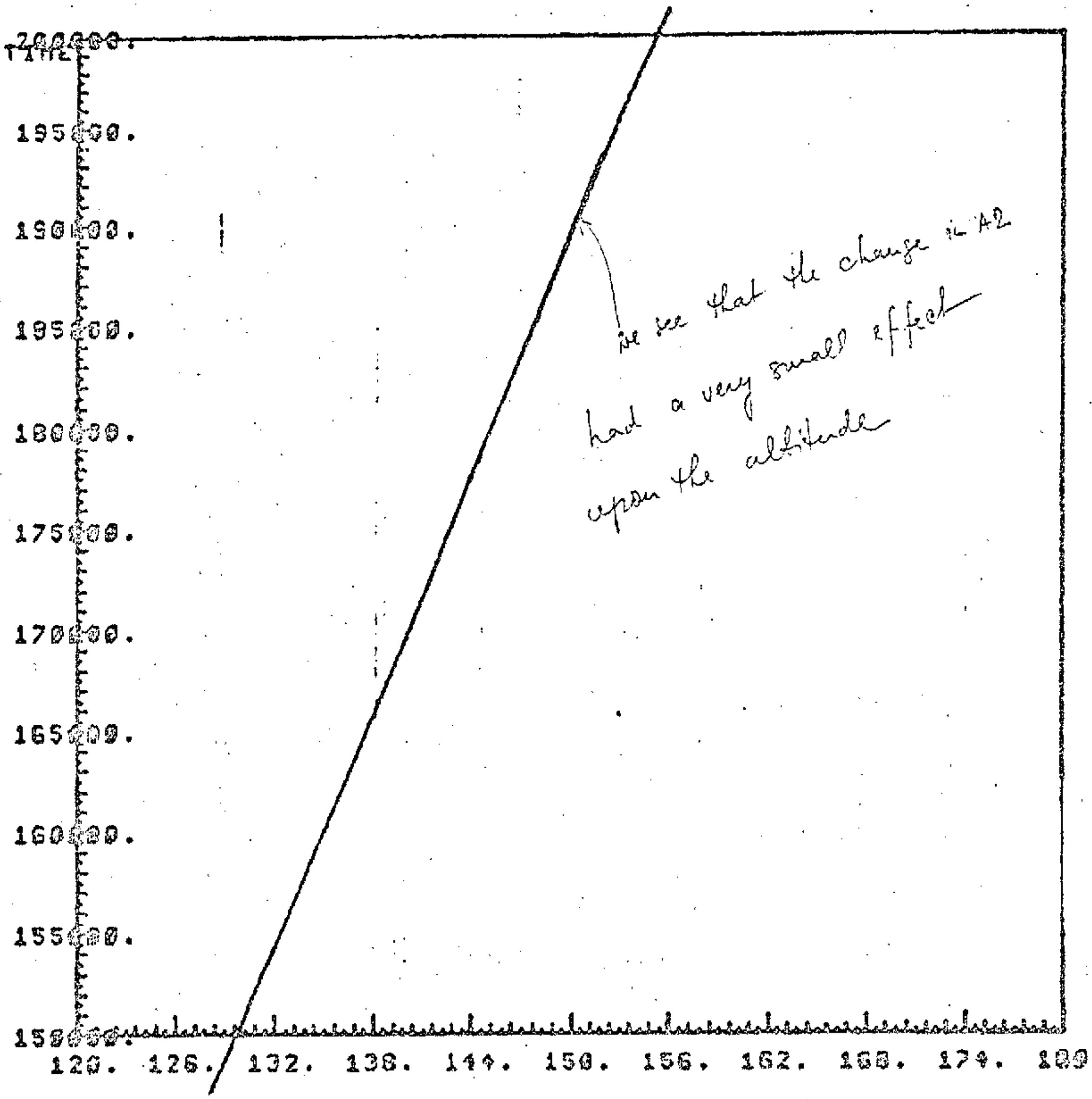
again using * display 10 * inertial velocity

The next display will take a closer look at the region where we stepped back in phase and changed A2 so we set ALT window at 150 000 → 200 000 feet, time window 120 → 180 sec and plot only the portion of the points which fall into this region.

13.- DISPLAY TTT(11,)&TIME

now we decide to display
the total THRUST which is stored
in TTT(11,) against TIME

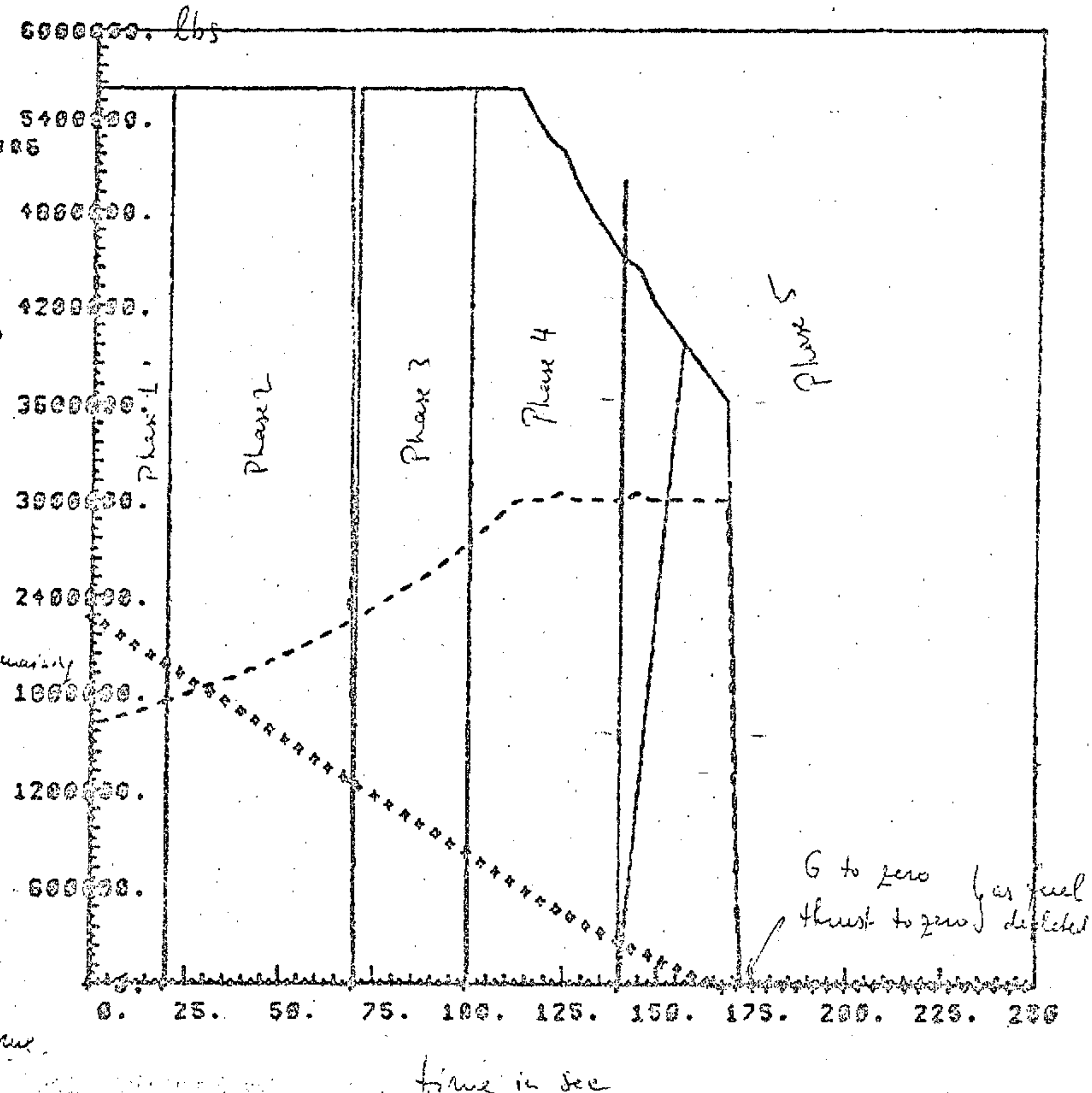
using the automatic scaling
of the display operator
and get the solid line
of the next picture.



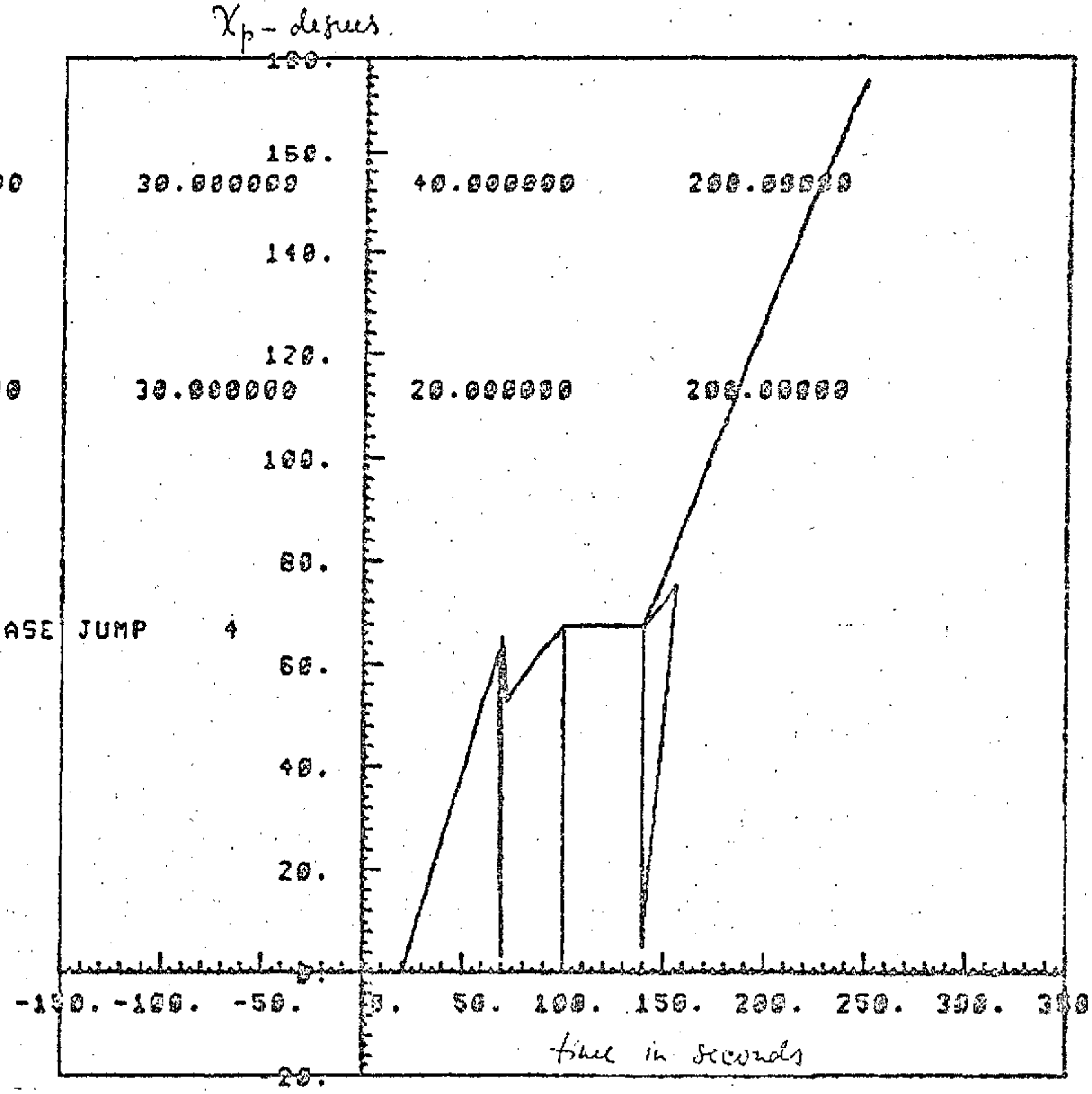
```
28.  [+]TTT(10,)
     [-]TTT(8,)*1E6

     + IS MFUEL
     - IS G-FORCE*10**6

     SOLID LINE IS
     THRUST

     NOW LOOK AT CHIP

     DISPLAY TTT(9,)
```

on the same picture
we display using ~~dotted~~ ++++ lines
TTT(10,) which is total fuel remaining
and using ---- lines
G-force ratio multiplied by 10^6

on the next display
look at X_p plotted against time.



```
 15.  PRINT TPHASE

NCO(TPHASE)=     5
TPHASE=
  20.000000       50.000000       30.000000       40.000000       200.00000
 16.  TPHASE(4)=20

 17.  PRINT TPHASE

NCO(TPHASE)=     5
TPHASE=
  20.000000       50.000000       30.000000       20.000000       200.00000
 18.  A2=A2/2

 19.  PRINT A2/PI*180

 .50000000
 20.  CALL REPHAS

WRITE 1 2 3 4 OR 5 FOR PHASE JUMP    4

 21.  PRINT INPHASE

INPHASE=  4.0000000
 22.  TSTOP=180

 23.  PRINT T

T=  100.00000
 24.  CALL FLY
```



Here the user decides to step back to phase 4 and shorten it to 20 seconds duration. So he prints TPHASE to see that currently phase 4 lasts 40 seconds, resets duration of fourth phase and slows down control turn to 0.5 degrees per second in phase 5. As a check 21. prints INPHASE confirming the step back.
To shorten the coast period he sets Tstop = 180 and as another check checks that T is set back to 100 seconds for the start of phase 4 [illegible]

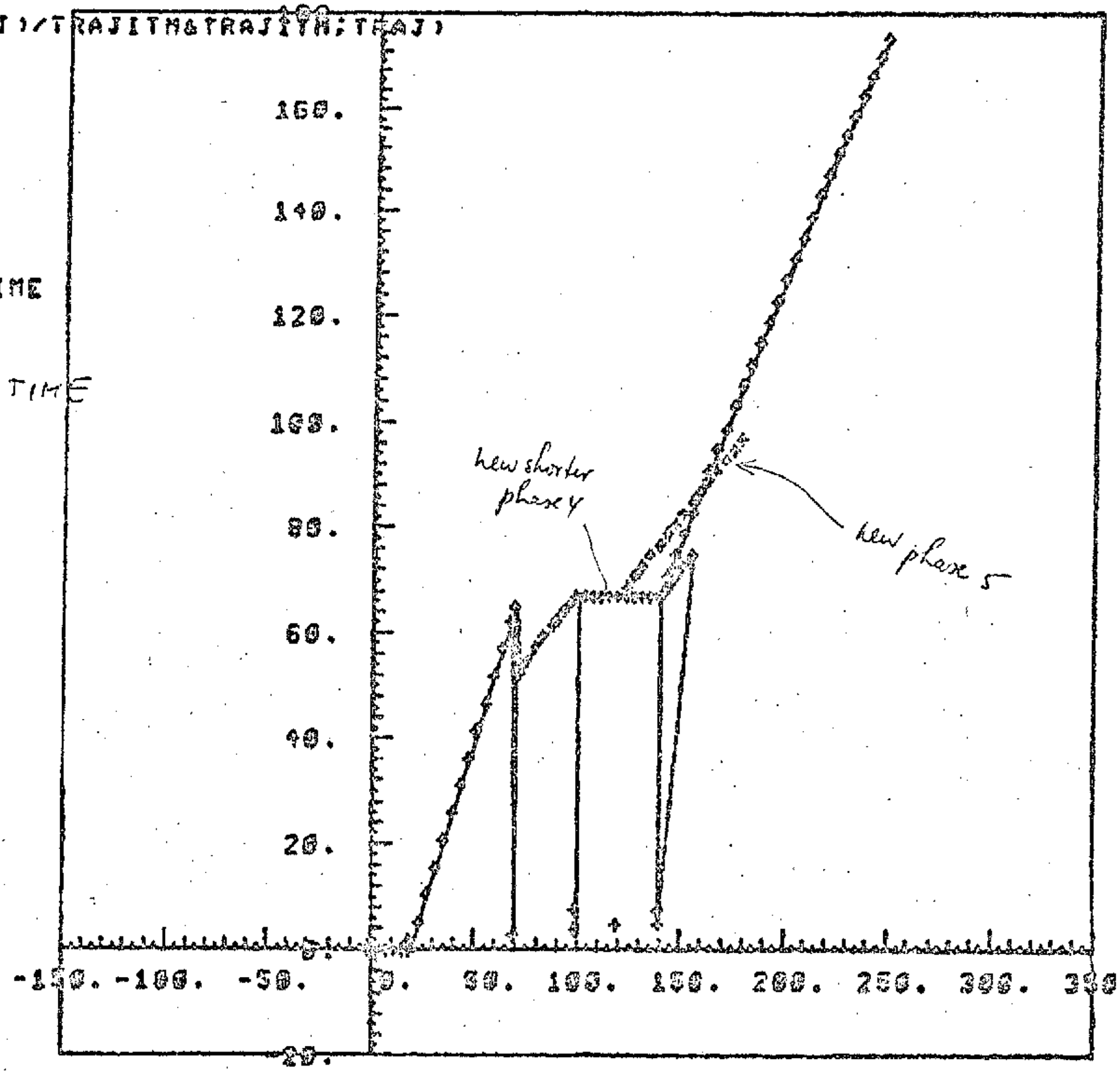
```
4000. STATEMENTS EXECUTED.TYPE GO OR QUIT.. GO
T=  180.00000
TSTOP=  180.00000
G=  2.7405646
GMAX=  3.0000000
Q= 0.
QMAX=  100.00000
ALT=  221752,33
ALTMAX=  300000.00
MFUEL=  820700.00
NCO(TRAJ)=  104   12
TRAJ=
 0.           0.           0.           0.           0.
 0.           0.           0.           0.           0.
 0.           0.

 1.0000000    1.0000000    1.0000000    1.0000000    1.0000000
 1.0000000    0.           1.0000000    1.0000000    1.0000000
 1.0000000    1.0000000

 874.47205    0.           1016.6676    0.           0.
0.            0.           1.5000000    0.           2277400.0
 5600000.0    .36379788E-11

 874.45131    79.061470    1016.6551    3497.8739    157.64779
 4066.6536    4.0000000    1.6270078    0.           2219132.0
 5600000.0    79.405444

 874.42807    161.67249    1016.6176    6995.6618    639.87404
 8133.2872    8.0000000    1.6551065    0.           2160064.0
 5600000.0    162.36044

 874.37534    247.96032    1016.5551    10493.270    1461.9209
 12199.661    12.000000    1.6841113    0.           2102596.0
 5600000.0    248.99229

 874.30014    330.65097    1016.4677    13990.636    2636.0048
```

user types GO interactively to continue. The statement limit is built into Sigma to avoid infinite loops and it can be set by the user to any convenient limit.

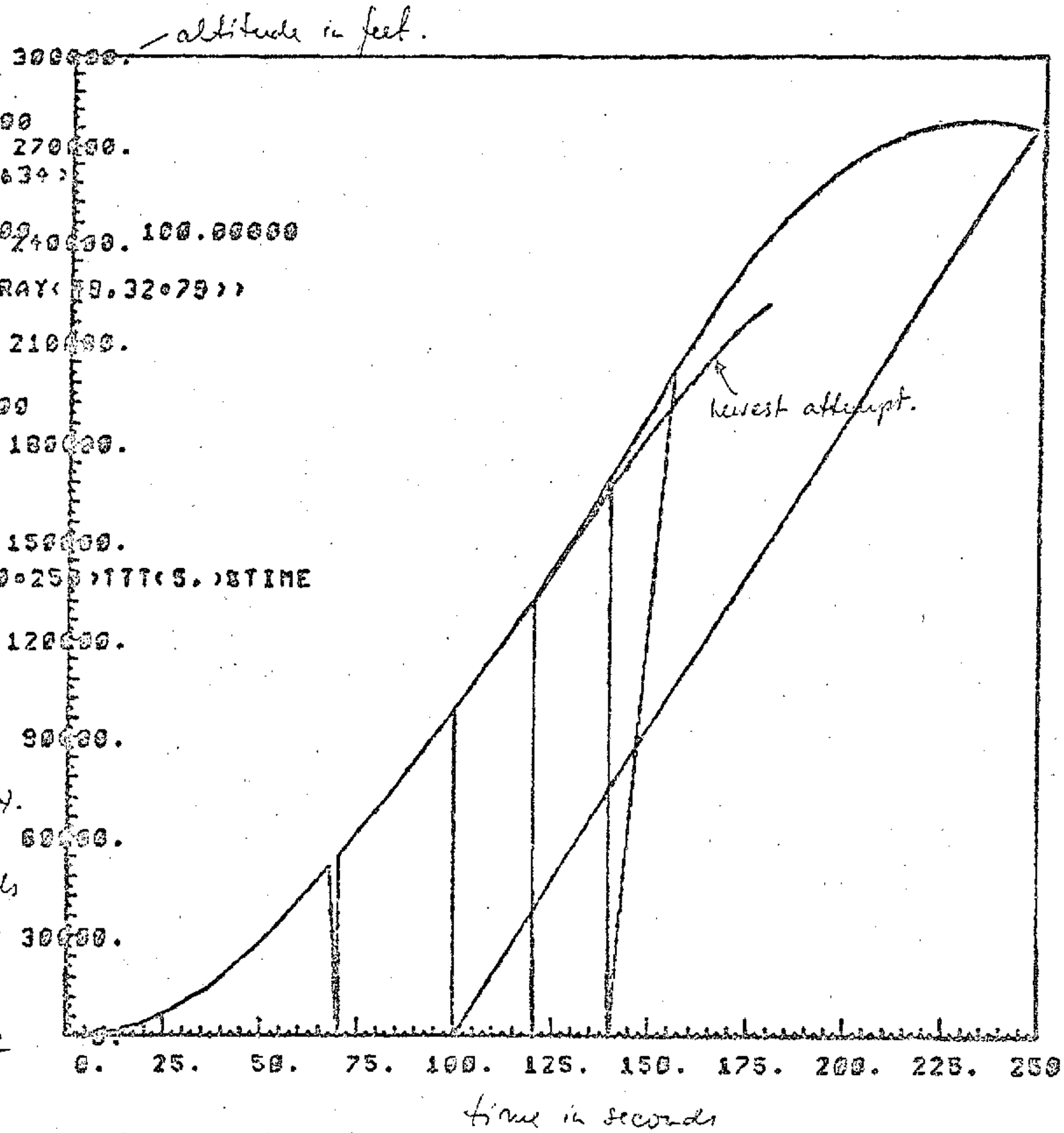
Fly signals the end of the calculation at T=180 sec and trajectory is printed.

```
26.   TTT=ARRAY(NCO(TRAJ)/TRAJITM&TRAJITM;TRAJ)
27.   TTT=TP(TTT)
28.   TIME=TTT(7,)
29.   [◊]TTT(9,)%TIME

      DISPLAY TTT(5,)%TIME
```

here the user regenerates TTT and TIME and plots the new x_p on top of the old curve showing that there is a new slower rising curve.



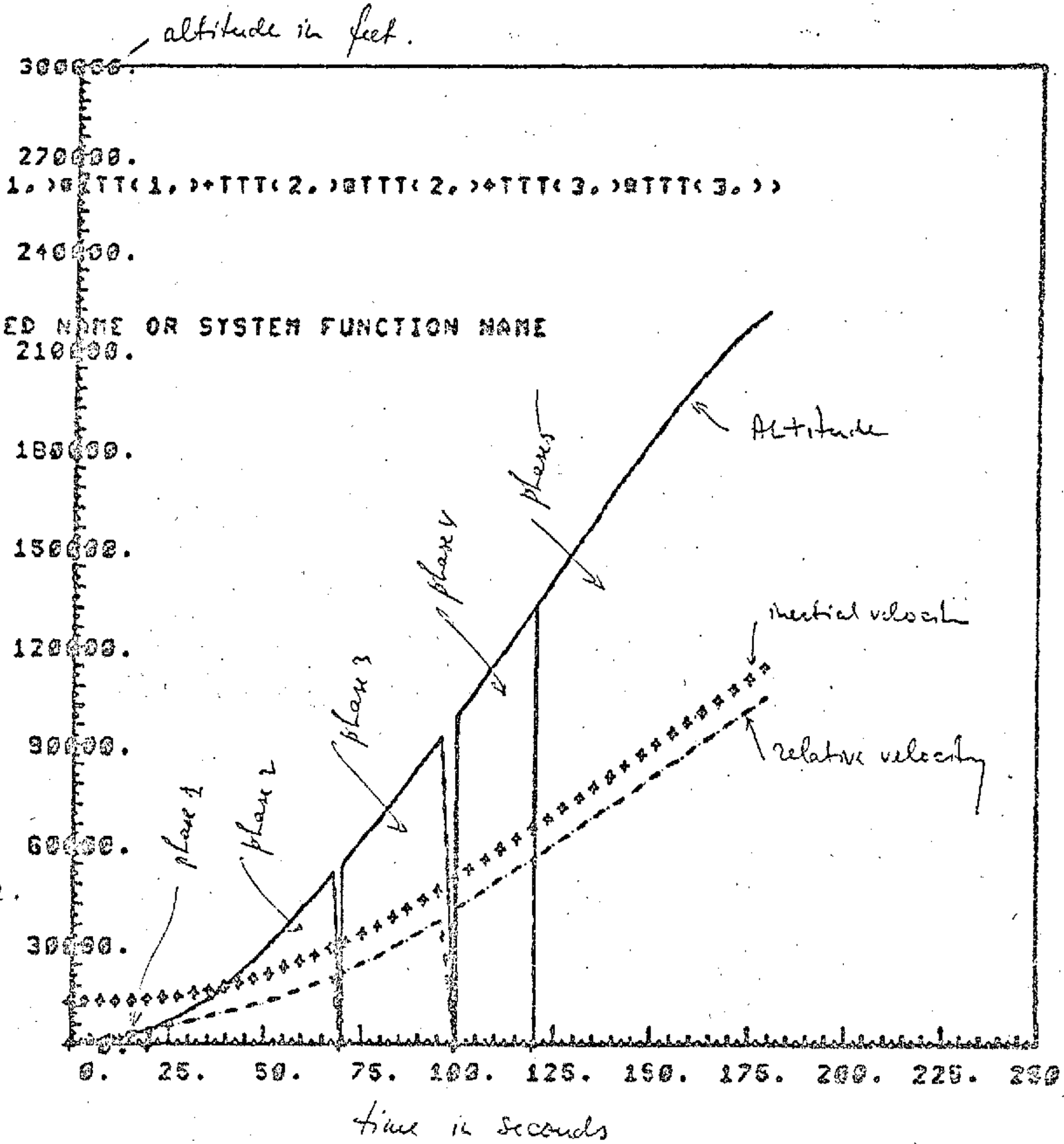
```
37.  PRINT NCO(TTT)

12.000000        104.00000
38.
     PRINT TTT(7,30&32&34)

92.000000        100.00000        100.00000
39.
     TTT=DROP(TTT,0,ARRAY([illegible]9,32∘79))

40.  PRINT NCO(TTT)

12.000000        56.000000
41.  TIME=TTT(7,)

42.  PRINT NCO(TIME)

56.000000
43.  DISPLAY(0∘300000&0∘250)TTT(5,)&TIME
```



The trajectory is now getting somewhat messy so it is time to discard uninteresting portions of it.

In 37, 38 above the user finds at which row phase 4 began in the old trajectory; this is row 32
The user also remembers that before the last addition TTT had 79 rows
(see page 6-14 and remember that TTT is the transpose of the array that is displayed there).
Hence in (39) the user uses the DROP operator of sigma to remove rows 32→79 from TTT

```
48.  [.-]TTT(12,)@10

     VELINRT=SQRT(TTT(1.)@TTT(1.)+TTT(2.)@TTT(2.)+TTT(3.)@TTT(3.))

49.  [+]VELIRT@10

@@@@@@@@ VELIRT @UNDEFINED NAME OR SYSTEM FUNCTION NAME

  1.  [+]VELINRT@10
```



here user once again has a clean trajectory and he decides to display inertial and relative velocity.

Sigma complains that he misspells VELINRT

So the user retypes it correctly.

Since the user is tired he now saves the workspace with

!LOGOUT

and leaves it stored for the next session.

7.0 REFERENCES

1) A Payload Sizing Ascent Trajectory Optimization Program (PSATOP) Langley Working Paper LWP-905 NASA Langley Research Center (1970)

2) Notes on a man-in-the-loop trajectory optimization program Private communication R. G. Toelle, NASA Marshall Space Flight Center (1972)

3) SIGMA, A New Language for Interactive Array-Oriented Computing R. Hagedorn, J. Reinfelds, C. E. Vandoni and L. Van Hove, CERN 73-5 CERN, Geneva, Switzerland (1973)

APPENDIX A

Alphabetical list of all names used by MILTOP.

ALT	h	feet	altitude of spacecraft $ALT = \sqrt{x^2+y^2+z^2} - REARTH$
ALTMAX		feet	altitude limit for automatic stopping of FLY routine
AREANOZ	A_e	ft^2	exit area of one engine for thrust calculations
AREAREF	S	ft^2	reference area for drag calculations
AZIM	A_z	radians	azimuth angle of launch site at launch time $t=0$
A1	a_1		in phase 5 $\chi_p = a_1 + a_2$ (time − time of start of phase)
A2	a_2		
A12	a_{12}		matrix elements of the matrix A which converts geocentric inertial coordinates to plumbline coordinates.
A22	a_{22}		
A32	a_{32}		
CA(ALT)	C_A	function	calculates drag coefficient in F_{AA}
CNP(ALT)	C_N'	function	calculates drag coefficient in F_{AN} not included as it is not required if $F_{AN} = 0$
CHIDOT	$\dot{\chi}$	rad/sec	rate of change of χ_p during phase two
CHIP	χ_p	rad.	pitch angle of the vehicle centerline w.r. to plumbline coords.
CHIY	χ_y	rad.	yaw angle of the vehicle centerline w.r. to plumbline coords
COSCHIP			$\cos(\chi_p)$ to avoid repeated cosine calls in NEWRHS
DENSITY	ρ	slug/ft^3	density of the atmosphere at the current altitude
DW	δw	ft/sec	step in the equations of motion in NEWRHS
DU	δu	ft/sec	
DV	δv	ft/sec	

F	f		flattness of the earth ratio = 1/298.3
FAA	~~lbs~~ F_{AA}	lbs	axial drag force
FAN	F_{AN}	lbs	normal drag force
FLY	macro		main routine to advance the trajectory
G			the current G-force measured in units of g_0
GMAX			maximum permissible G-force. At G ≥ GMAX automatic throttling begins to hold G-force at GMAX.
GRAVIT	μ_e / r^3		product of gravitational constant by mass of earth divided by the cube of the radius vector in geocentric coordinates
GZERO	g_0	ft/sec^3	sea level gravitational acceleration g_0 = 32.1740476 ft/sec
H	h	sec	step size of integration of equations of state (mathematical name clashes with mathematical symbol for altitude but both are never used on the same section).
I			Do-loop index in FLY
INI	macro		initializes first run of a problem
INSTAG1	macro		initializes all constants associated with the first stage of a vehicle.
INSTAG2	—"—		to be provided by the user for stage 2
INPHASE	integer		holds the number of current phase of the flight 1, 2, 3, 4 or 5
ISP	$ISP_{vac.}$ $ISP_{s.l.}$	sec	impulse of vehicle stage should match the thrust specification so that if ~~\$~~ THRENG is a sea-level thrust ISP should be a sea-level impulse and if THRENG is a vacuum thrust, ISP should be a vacuum impulse.

K1 K2 K3 K4			auxiliary variables for Runge-Kutta integration.
LAT	θ_0	rad	geodetic latitude of the launch site.
MACH	M		Mach number $M = V_{REL}/a$ where a is speed of sound.
MDOT	$\dot{m}$	lbs/sec	rate of mass flow of propellant
MFUEL		lbs	mass of propellant remaining on board.
MLBS	W	lbs	mass of vehicle including payload and propellant
MUGRAV	μ_e	feet^3/sec^2	product of univ. grav. constant and earth mass $1.407656 \cdot 10^{16}$
NENG			number of engines in this stage
NEWCHI	macro		calculates χ_p χ_y for this point on the trajectory
NEWDRAG	macro		calculates F_{AA} F_{AN} —"—
NEWRHS	macro		calculates new value of all right-hand sides of eqns. of state
NEWTHRS	macro		calculates new value of the total THRUST
NOFUEL	macro		resets MDOT, THRUST and MFUEL to allow coasting after exhaustion of propellant.
NOWSTEP			counts successive steps on the trajectory for the purpose of saving only each savestep'th step.
NSTATE	vector		auxiliary vector from which SETVAR sets all individually named variables. Required because Runge Kutta integration takes three partial steps from the same start before making the full step to $t \to t+H$ so that STATE vector must be preserved unchanged.

OMEGA	Ω_e	rad/sec	angular velocity of the earth 7.292115×10^{-5} rad./sec
PHASE	5×11 array		stores the state of the flight at the beginning of each phase so that REPHAS can step back to any previously entered phase.
PRESS	function (not included)		looks up atmospheric pressure from atmosphere table.
Q	q	lbs/ft^2	dynamic pressure
QMAX			dynamic pressure limit which causes FLY to stop.
REARTH	R_e	feet	Radius of the earth 20.925732×10^6 feet for a spherical earth ($R(\phi)$ for an ellipsoid earth).
REPHAS	macro		returns to a user selected previously entered phase
RHS	vector		vector in which NEWRHS returns the value of the right-hand sides of the equations of state.
RUNGE	macro		integrates the equations of state by one step $t \to t+H$
SAVPHAS	macro		saves the first point of each phase in the array PHASE
SAVTRAJ	macro		saves user selected quantities in the trajectory vector TRAJ
SAVSTEP	scalar		FLY will save each SAVSTEP'th step in TRAJ
SCR SCRACH any other name starting SCR			Scratch variables used very locally to hold a quantity for a short while. Reused in several macros.
SETVAR	macro		sets all individually named variables from NSTATE and calculates all variables that depend on them.
SIGMA	σ	rad.	angle between $\vec{F}_{AN}$ and the $\hat{\rho}$ axis in the auxiliary coordinate system where $\vec{F}_{AN}$ is in the direction of the $\hat{\phi}\hat{\rho}$ plane component of $\vec{V}_{Rel}$ of the vehicle.

SINCHIP		$\sin(\chi_p)$	to avoid repeated sin function calls in NEWRHS
STATE		vector	contains current state vector (w u v x y z ¥)
T		sec	time since launch
THETA	θ_1	rad	$\theta_1 = \pi/2 - \theta_0$ where θ_0 is geodetic latitude LAT
THRENG	F_i	lbs	thrust per engine either nominal sea-level thrust or vacuum thrust.
THRUST			total thrust of vehicle constant unless throttled
TPHASE		vector	vector holding the duration of each phase in sec.
TPHSTOP		sec	time at which current phase ends and SAVPHAS has to be called to start the next phase
TRAJ		vector	holds accumulated trajectory flown so far.
TRAJIT		scalar	set to number of items accumulated in trajectory at each step by SAVTRAJ
TRAJSHO		macro	displays the trajectory
TSEALEV		control variable	to indicate if THRENG is sea level or vacuum thrust for possible use in NEWTHRS (not used at the moment).
TSTOP		sec	time at which FLY stops. ie. the user-selected next decision point which may be anywhere on a trajectory.
U	u	feet/sec	plumbline coord. y-component of velocity of vehicle
UREL	$\left\{ \begin{matrix} u \\ u_{rel} \end{matrix} \right.$	feet/sec	—"— —"— relative velocity (rel w.r.t. to atmosphere)
V	v	feet/sec	—"— z-component of velocity of vehicle
VREL	$\left\{ \begin{matrix} v \\ v_{rel} \end{matrix} \right.$	feet/sec	—"— —"— relative velocity
VELREL	V_R	feet/sec	magnitude of relative velocity = $\sqrt{WREL^2 + UREL^2 + VREL^2}$

W	w	feet/sec	plumbline coordinate	x-component of velocity
WREL	$\begin{cases} \underline{w} \\ w_{rel} \end{cases}$	feet/sec	—"—	—"— relative velocity
X	x	feet	plumbline	x-coordinate of vehicle
Y	y	feet	—"—	y-coordinate —"—
Z	z	feet	—"—	z-coordinate —"—

APPENDIX B

The Sigma Language.

In this brief descriptive summary of Sigma language features used in this report we assume that the user is familiar with FORTRAN so it suffices to say that in Sigma the assignment statements, IF statements and DO loops are similar to Fortran and perform in the same way.

Array definition is done by the operator ARRAY which

(i) with one argument as for example in ARRAY (100) defines a vector of 100 components all equal to 1.

(ii) with two arguments as for example in ARRAY (100, 0 # 1) defines a vector of 100 equally spaced component whose first element is 0 and last one is 1.

Array indexing is done by subscripting but

(i) A (5) is the fifth element of the array A

(ii) A (2 & 5) is the two component array consisting of the second and fifth element of A in that order A(2) A(5)

(iii) A (Array (5, 1 # 5)) is a five component array consisting of the first five components of A.

(iv) A (INPHASE,) "the missing index" denotes all elements of this dimension.

Concatenation (&) the "ampersand" is the concatenation operator in Sigma. It joins two arrays together end to end so that if

Z = ARRAY (10, 1 # 10) & ARRAY (3, 100 # 300)

then Z is the 13 component array 1 2 3 4 5 6 7 8 9 10 100 200 300

<u>Macro</u> is a piece of code which is executed at the point at which the macro is called. A macro does not "change the level of the program" hence all variables defined at macro call time are available to the macro body as if they were in a common block.